(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2004 年4 月29 日 (29.04.2004)

PCT

(10) 国際公開番号
WO 2004/036432 A1

(51) 国際特許分類[7]: G06F 12/00, 17/30

(21) 国際出願番号: PCT/JP2003/013443

(22) 国際出願日: 2003 年10 月21 日 (21.10.2003)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願 2002-306296
2002 年10 月21 日 (21.10.2002) JP

(71) 出願人 (米国を除く全ての指定国について): アネックスシステムズ株式会社 (ANNEX SYSTEMS INCORPORATED) [JP/JP]; 〒107-0061 東京都 港区 北青山 3丁目 7番 1号 Tokyo (JP).

(71) 出願人 および
(72) 発明者: 玉津 雅晴 (TAMATSU,Masaharu) [JP/JP]; 〒206-0023 東京都 多摩市 馬引沢 2丁目 14番 14号 サンセットヒルズII 403 Tokyo (JP).

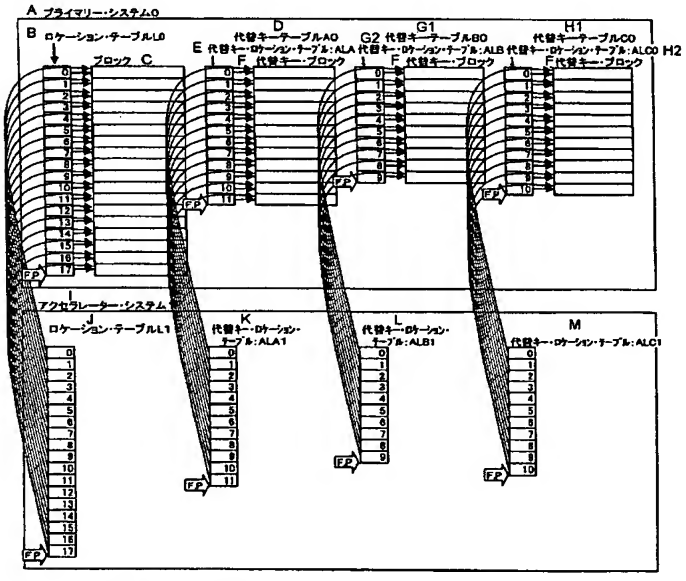
(74) 代理人: 山口 朔生, 外(YAMAGUCHI,Sakuo et al.); 〒101-0032 東京都 千代田区 岩本町 2-15-10 ニュー山本ビル 3階 Tokyo (JP).

(81) 指定国 (国内): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK,

[続葉有]

(54) Title: DATABASE ACCELERATOR

(54) 発明の名称: データベースのアクセラレーター



A...PRIMARY SYSTEM 0
B...LOCATION TABLE L0
C...BLOCK
D...ALTERNATIVE KEY TABLE A0
E...ALTERNATIVE KEY LOCATION TABLE: ALA
F...ALTERNATIVE KEY BLOCK
G1...ALTERNATIVE KEY TABLE B0
G2...ALTERNATIVE KEY LOCATION TABLE: ALB
H1...ALTERNATIVE KEY TABLE C0
H2...ALTERNATIVE KEY LOCATION TABLE: ALC0
I...ACCELERATOR SYSTEM
J...LOCATION TABLE L1
K...ALTERNATIVE KEY LOCATION TABLE: ALA1
L...ALTERNATIVE KEY LOCATION TABLE: ALB1
M...ALTERNATIVE KEY LOCATION TABLE: ALC1

(57) Abstract: A database accelerator can assure scalability in a database processing by stepwise and at a low cost and provide scalability thousands times greater. For a primary system in which data update is performed, the accelerator has a copy of a location table equivalent to a main key index and a copy of an alternative key location table equivalent to an alternative key index on an accelerator system. When the location table and the alternative key location table are modified in the primary system, the location table and the alternative key location table in the accelerator system are updated in synchronization. Data is present only in the primary system. Processing requests are distributed to the accelerator system so that they are processed in parallel, thereby assuring a great extension of the processing ability.

(57) 要約: データベース処理におけるスケーラビリティの確保を、段階的に安価に可能とする一方で、数千倍程度の大きなスケーラビリティを提供すること。 本方式は、データ更新が行なわれるプライマリー・システムに対して、主キー・インデックスに相当するロケーション・テーブルと代替キー・インデックスに相当する代替キー・ロケーション・テーブルのコピーをアクセラレーター・システム上に保有する。プライマリー・システムでロケーション・テーブル、代替キー・ロケーション・テーブルに変更があった場合は、アクセラレーター・システムのロケーション・テーブル、代替キー・ロケーション・テーブルの更新を同期して行う。データはプライマリー・システム上にのみ保有する。処理要求をアクセラレーター・システムに分散し、並列処理を行うことにより、処理能力の大きな拡張性を確保する。

WO 2004/036432 A1

DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) 指定国 (広域): ARIPO 特許 (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア特許 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ特許 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI 特許 (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明 細 書

データベースのアクセラレーター

発明の属する技術分野

本発明はコンピューターにおけるデータ格納・検索方式及びシステムに係り、特にデータ格納および検索に対して、大きなスケーラビリティ（処理能力の拡張性）を与えることを目的とするものである。スケーラビリティとは、処理能力の拡張性のことである。単一のデータベース・システムが、ある一定の性能を有している場合に、その性能に対して、数倍から数千倍以上の処理能力を提供するもので、データベース・システムの性能向上に関するものである。

従来の技術

従来のコンピューターによるデータベース格納検索方式は、「Jeffrey D. Ullman 著、国井他１名訳、″データベース・システムの原理″，第１版，日本コンピューター協会，1985 年 5 月 25 日, p45-71」、「Samuel Leffler 他著/中村明他訳,” UNIX4. 3BSD の設計と実装”, 丸善株式会社，1991 年 6 月 30 日, p193-191」及び「Michael J. Folk 他著/楠本博之訳,″bit 別冊 ファイル構造″, 共立出版 株式会社, 1997 年 6 月 5 日, p169-191」に記載されているように、基本的には階層型のインデックスを使用するものであった。

本発明者は、従来の階層型インデックスに代えてロケーション・テーブルと代替キー・テーブルという概念を導入し、インデックスの処理に伴う複雑な処理を簡素化し、テーブル自体の検索をバイナリー・サーチなどの手法を用いることにより、高速化と、メンテナンスの容易性を確保できるようにした「データ格納検索方式」を発明した（日本国特許第３３４５６２８、米国特許第６４１５３７５号参照、米国特許第６５８４５５５号参照）。

更に、「データベースの再編成システム、並びに、データベース（以下、「データベース再編成システム」と呼ぶ）：ＰＣＴ／ＪＰ０３／１１５９２」では、「データ検索格納方式」で発明したデータベースに対して、データベースを稼動させながら再編成

が行える仕組を提案した。更に、代替キー・テーブルに対して代替キー・ロケーション・テーブルを付加することにより、効率的な再編成が可能となることを提案している。

ここで、本発明者が提案したデータ格納検索方式の内容について簡単に説明しておくことにする。本発明のデータ格納検索方式は、ロケーション・テーブルと代替キー・テーブルを使用し、それらに対して、バイナリー・サーチを行うことにより、目的のレコードを検索するものである。

データレコードはプライマリー・ブロックに、主キーの順に格納する。そのプライマリー・ブロックにデータレコードを追加する場合に、プライマリー・ブロックが満杯である場合に、オーバーフロー・ブロックを、そのプライマリー・ブロックに連結して、データレコードを格納する。オーバーフロー・ブロックに、更にオーバーフロー・ブロックを連結することが可能である。

各プライマリー・ブロックのアドレスが記載されたロケーション・テーブル・レコード（または、ロケーション・テーブル・エントリー）を連続領域に有するロケーション・テーブルを有する。

ロケーション・テーブルは連続した領域に予め確保する。ここで連続した領域とは、論理的な順序であり、物理的な領域では、離れていても良い。このような場合には、アドレス変換テーブルを用いて、論理的に連続していると扱うことができる。これは、以下の説明でも同様である。

ロケーション・テーブルの使用領域の最後を示すために、最終ポインターを用いる。

レコードはブロックと言う固定長の格納領域に格納する。ブロックは、プライマリー・ブロックとオーバーフロー・ブロックからなる。最後のプライマリー・ブロックにレコードが追加できない場合は、その後にプライマリー・ブロックを追加して、レコードの格納を行う。

ここで、連結とは、物理的に連結されていることを意味するのではなく、プライマリー・ブロックが一番目のオーバーフロー・ブロックのアドレスを保持し、１番目のオーバーフロー・ブロックが２番目のオーバーフロー・ブロックのアドレスを保持している状態が、あたかも物理的にブロックが繋がれているように扱えることから、そのような表現を使用している（以下、同様である）。

このように格納するので、ロケーション・テーブル・エントリーは、主キーの順番に並んでいる。主キーによる検索は、ロケーション・テーブルの最初のアドレスと最終ポインターが指しているロケーション・テーブル・エントリーの間に対して、バイナリー・サーチを行うことにより、ブロックを見つけ、そのブロック内で目的のレコードを見つける。当該ブロックにオーバーフロー・ブロックが連結されている場合は、オーバーフロー・ブロックも検索の対象となる。

ここでは、検索に関して述べているが、レコードの更新、追加、削除も同様のロジックで実現できる。

代替キー値と主キー値からなる代替キー・レコード（または代替キー・エントリー）は、代替キー・ブロックに代替キー値の順番に格納する。

その代替キー・ブロックに代替キー・エントリーを追加する場合に、その代替キー・ブロックが満杯である場合に、代替キー・オーバーフロー・ブロックを代替キー・ブロックに連結して、代替キー・エントリーを格納する。代替キー・オーバーフロー・ブロックに、更に代替キー・オーバーフロー・ブロックを連結することが可能である。

各代替キー・ブロックのアドレスが記載された代替キー・ロケーション・テーブル・レコード（または、代替キー・ロケーション・テーブル・エントリー）を連続領域に有する代替キー・ロケーション・テーブルを有する。

代替キー・ロケーション・テーブルは連続した領域に予め確保する。

代替キー・ロケーション・テーブルの使用領域の最後を示すために、代替キー最終ポインターを用いる。

代替キー・エントリーの追加で、既存の代替キー・エントリーの代替キー値より大きい代替キー値を持つ代替キー・エントリーは、最後の代替キー・ブロックに格納し、その代替キー・ブロックに格納できない場合には新たに代替キー・ブロックを作成し、その代替キー・ブロックに当該レコードを格納する。

代替キー・ロケーション・テーブルと代替キー・ブロックを組にして、代替キー・テーブルと呼ぶ。

代替キーは、データベースにおけるノンユニークなキーのことで、例えば、従業員

データベースにおける、氏名や生年月日などのことである。代替キーは、或る種類のデータベースに対して、無くても良いし、複数あっても構わない。

或る代替キーを持ったレコードを検索する方法は、代替キー・ロケーション・テーブルの最初のエントリーと、代替キー最終ポインターが指している代替キー・ロケーション・テーブル・エントリーの間をバイナリー・サーチし、目的の代替キー・ブロックを見つけ、その代替キー・ブロックの中を検索して、目的の代替キーを持つ代替キー・エントリーを見つける。その代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、代替キー・オーバーフロー・ブロックも検索の対象となる。

次に、その代替キー・エントリーの主キーによって、ロケーション・テーブルをバイナリー・サーチし、目的のブロックを見つけ、そのブロック内から目的のレコードを見つけ出す。当該ブロックにオーバーフロー・ブロックが連結されている場合は、オーバーフロー・ブロックも検索の対象となる。

尚、代替キーはノンユニークであるので、同一の代替キー値を持ったレコードが複数存在する可能性がある。この場合は、代替キー・ブロック中の次の代替キー・レコードが同一の代替キー値である場合には、上記と同様な動作を繰り返す。

ここでは、検索に関して述べているが、レコードの更新、追加、削除も同様のロジックで実現できる。

また、複数の代替キーが存在する場合には、代替キーの種類と同じ数の代替キー・テーブルを作成し、使用することになる。

以上のような特徴を持つ「データ格納検索方式」では、ハードウエアの性能を最大限に引き出し、大幅な高速化を実現することが可能であるが、一方で、その性能の上限になると、それ以上の処理性能を提供することができない、つまり、スケーラビリティが無い方式であった。これは、従来の他の方法も本質的には同様である。但し、従来の方法はハードディスクの使用を前提に考えられているため、これを、半導体などの高速メモリー上に乗せることにより、ハードディスク上で実現しているスピードに対して、高速化を図ることが可能であったが、高速メモリーの性能限界が上限になるものであり、本質的なスケーラビリティを提供するものではなかった。

従来の技術でスケーラビリティを向上させる技術とされるものに、ロード・バランサーが上げられる。これは、サーバーを複数用意して、それが見かけ上一台のサーバーであるかのようにするものである。外部からの処理要求が過大な場合、まず、ロード・バランサーに処理要求を入れ、ロード・バランサーが処理要求を複数のサーバーの何れかに割り当てることにより、一台当りの処理負荷を低減させ、それによりスケーラビリティの向上を図ろうとするものである。しかしながら、この方式には致命的な欠陥があった。それは、複数のサーバーが処理できるのは、処理のロジック部分のみであり、データベースは一つであり、各サーバーから同一のデータベースをアクセスするため、データベースの性能が処理性能を規定してしまうということであった。つまり、処理要求が増加して、サーバーを増加させても、データベースの性能が上限となってしまうということである。

「データ格納検索方式」においてスケーラビリティを確保する方法の一つとして、本発明者による「データバックアップ・リカバリー方式ＰＣＴ／ＪＰ０１／０３１２６」で、実際のデータ更新・参照用のプライマリー・システムに対して、そのバックアップ・コピーであるセカンダリー・システムを設け、本来的な目的としてはバックアップ・リカバリー用のシステムとして利用する他に、セカンダリー・システムを参照用のシステムとして使用することにより、一定のスケーラビリティが確保できることが考案されている。

この方法では、更新トランザクション量が、参照トランザクション量に比較して、一般的に１対１０程度であることから、セカンダリー・システムを参照用に使用することにより、プライマリー・システムの負荷を軽減することが可能であるとするものである。しかしながら、セカンダリー・システムを更新用には使用できないことから、スケーラビリティの確保としては限界が低いものであった。

また、この方法では、セカンダリー・システムは、プライマリー・システムと同等の構成が必要であり、ロケーション・テーブル、データ格納用ファイル（ブロックの集合）、代替キー・テーブル（ゼロ、または、１つ、または、複数）をセットとして用意するものであったため、バックアップの目的以外にスケーラビリティ確保の目的

として、セカンダリー・システムを複数用意することは、費用的に負担が大きくなるものであった。

「データ格納検索方式」以外の従来の方法では、上記以外に、データを保持するサーバーのミラー化によって性能向上を図ることが一般的に行われているが、ミラーとなるデータベースは、元のデータベースと同じ容量が必要であるため、大きな格納容量を必要とするものである上に、ミラー・データベースは基本的に参照用にしか使用できず、更新は制限が大きなものであった。主サーバーのデータが変更された場合に、ミラー・サーバーに、その更新内容を反映することが、一定時間遅れてしか実現できないという制限である。これは、ミラー・サーバーでデータの更新を行うと、主サーバーでの更新との時間差（タイムラグ）のため、更新結果が無効になってしまう危険性があるということであり、通常のデータ処理では採用できない方法であった。このように、ミラー・サーバーを使用した性能向上は、通常のリアル・タイム・データ処理では使用できないものであった。

このように、データベースに対するスケーラビリティの確保は重要な課題であるが、十分な解決方法が存在してこなかったのが実情である。これは、従来の方法では、データそのものを複写する必要があるのに加え、インデックス構造が複雑なために、インデックスが更新された場合の複写が面倒であった、ということに起因している。

発明が解決しようとする課題

処理量が増加しても、それに追随してスケーラビリティが確保できるデータベースは、情報処理において重要な要求であった。このような要求にこたえるのが、本発明である。

課題を解決する為の手段

本発明は、主キーを含むデータ項目を持つデータレコードと、データレコードを主

キーの順に格納するプライマリー・ブロックと、各プライマリー・ブロックのアドレスが記載されたロケーション・テーブル・エントリーを連続領域に有するロケーション・テーブルとを保有するプライマリー・システムを備え、及び、各プライマリー・ブロックのアドレスが記載されたフランド・ロケーション・テーブル・エントリーを連続領域に有するフランド・ロケーション・テーブルを保有するアクセラレーター・システムを備えたデータベースのアクセラレーターにある。

また、本発明は、主キーと代替キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、代替キーと主キーからなる代替キー・エントリーと、代替キー・エントリーを含む代替キー・ブロックと、代替キー・ロケーション・テーブル・エントリーを連続領域に有する代替キー・ロケーション・テーブルを保有するプライマリー・システムを備え、及び、フランド代替キー・ロケーション・テーブル・エントリーを連続領域に有するフランド代替キー・ロケーション・テーブルを保有するアクセラレーター・システムを備えたデータベースのアクセラレーターにある。

本発明の概要は、次のようなものである。「データ検索格納方式」を用いてデータベース・システムを構築し、主キー（異なるレコードのキー値が重複しないユニークなキーで、レコードに一つ存在するもの）での検索の場合を考えてみる。主キーの検索は、ロケーション・テーブルに対してバイナリー・サーチを用いて行うが、この場合のアクセス回数は、以下のようになる。

仮に、データ件数が、１０億件ある場合で、ブロックに平均して２０レコード格納できるとすると、ブロックの数は５０００万件となる。オーバーフロー・ブロックが発生していない場合、ロケーション・テーブル・エントリーの数が５０００万件となる。このロケーション・テーブルに対してバイナリー・サーチを行って、対象となるブロックを検索することになるが、バイナリー・サーチの回数は、底を２とする１ｏｇの値となる。この場合は、２５．６となり、切り上げて２６回以内が解となる。

一方、ブロック内におけるレコードの検索は、特殊な方法を使わずに、先頭からレコードを読んでいった場合で、期待値は１０回となる。また、バイナリー・サーチが可能な場合は５回以内となる。

つまり、一つのブロックを検出するために、ロケーション・テーブルを２６回検索

する必要があることを示している。また、レコードを検索するための負荷は、ロケーション・テーブルに対するアクセスの方が、ブロックに対する負荷よりも高いことを示している。

このことは、代替キーを用いた検索でも同様である。本発明では、代替キー・テーブルに、代替キー・ロケーション・テーブルを採用した形式を用いることを前提に説明を行う。代替キーを用いた検索は、代替キー・ロケーション・テーブルに対してバイナリー・サーチを行うことによって行う。代替キー・ブロックが平均的に５０のエントリーを保持しているとする。代替キー・ブロック数は２０００万ブロックとなる。つまり、代替キー・ロケーション・テーブル・エントリーの数が２０００万ということになる。この代替キー・ロケーション・テーブルに対してバイナリー・サーチを行う。バイナリー・サーチの回数は、２４．３となり、切り上げて２５回以内となる。

代替キー・ブロック内の代替キー・エントリーは固定長とすることができるので、代替キー・エントリーをバイナリー・サーチを用いて検索する場合は、６回以内となる。

つまり、代替キーロケーション・テーブルに対する負荷が、代替キー・ブロックに対する負荷を大きく上回ることになる。

また、同一のブロックをアクセスする確率は、かなり低いものである。何故ならば、トランザクションの平均時間が１００分の１秒として、１秒あたりのトランザクション処理数が１万件だとし、トランザクションあたりのデータ更新件数が５０件とすると、以下のようになる。

あるトランザクションが実行中に、他のトランザクションが実行されるのは、平均的に１００トランザクションとなる。となると、その時点で同時に更新が行われるレコードは、１００×５０＝５０００となる。

一方、ブロックの総数は５０００万ブロックであるから、ブロック排他を前提とした場合でも、同一のブロックに格納されているレコードが同時に使用される確率は、非常に低いものとなる。

このことは、ロケーション・テーブルや代替キー・ロケーション・テーブルを複数用意し、最新のエントリー、代替キー・エントリーをコピーして、キー検索の可能パス数を増加し、更に、各パスに対してＣＰＵを割り当てることにより、処理データ量を増加することができることを示している。

ここで、データベース（ロケーション・テーブルとブロック、代替キー・ロケーション・テーブルと代替キー・ブロック）を保持して検索・更新を行うサーバーをプライマリー・システムと呼ぶことにする。図１はプライマリー・システムの一例である。データを格納するブロックと、ブロックを管理するロケーション・テーブルの組、代替キー・エントリーを格納する代替キー・ブロックと、代替キー・ブロックを管理する代替キー・ロケーション・テーブルの組（代替キー・テーブル）とからなる。代替キー・テーブルは、代替キーの種類毎に作成する。代替キーの種類とは、社員マスターにおける、所属や生年月日、入社年月日などである。

図１では、代替キーの種類が３種類の場合を表している。

図２は、プライマリー・システムとアクセラレーター・システムの例である。アクセラレーター・システムとは、プライマリー・システムの機能の内、ロケーション・テーブルと代替キー・ロケーション・テーブルの全部または一部を保持して、それらのテーブルをプライマリー・システムと同期させる機能と、それらのテーブルを用いてレコードが格納されているブロックを検索する機能を保持しているものを言う。アクセラレーター・システムはブロックを保有せず、アクセラレーター・システムのロケーション・テーブル・エントリーは、プライマリー・システムのブロックのアドレスを保持している。同様に、アクセラレーター・システムの代替キー・ロケーション・テーブル・エントリーは、プライマリー・システムの代替キー・ブロックのアドレスを保有している。この図では、プライマリー・システムとアクセラレーター・システムの関係を具体的に示すため、アクセラレーター・システムのロケーション・テーブル及び代替キー・ロケーション・テーブルが、プライマリー・システムのブロック及び代替キー・ブロックを指している線を示したが、これより後の図面では、図面を簡潔にするため、アクセラレーター・システムからプライマリー・システムへの線は省

略している。

また、オーバーフロー・ブロックや代替キー・オーバーフロー・ブロックが存在するか否かは本質的な問題では無いので、ここを始め多くの図面では省略している。

プライマリー・システムで、データの追加・更新・削除などにより、ブロック内のデータが書き換えられ、それに伴ってロケーション・テーブルが変更された場合には、プライマリー・システムからアクセラレーター・システムに対して、ロケーション・テーブルの変更箇所（ロケーション・テーブル・エントリーの番号）と変更内容を送信して、アクセラレーター・システムの当該ロケーション・テーブル・エントリーの更新を行う。

同様に、プライマリー・システムで、データの追加・更新・削除などにより、ブロック内のデータが書き換えられ、それに伴って代替キーが変更された場合、代替キー・ブロック内の代替キー・エントリーの書き換えが行われるが、それに伴って代替キー・ロケーション・テーブルの更新が行われた場合には、プライマリー・システムからアクセラレーター・システムに対して、代替キー・ロケーション・テーブルの変更箇所（代替キー・ロケーション・テーブル・エントリーの番号）と変更内容を送信して、アクセラレーター・システムの当該代替キー・ロケーション・テーブル・エントリーの更新を行う。

このようにして、アクセラレーター・システムでは、最新のロケーション・テーブル及び代替キー・ロケーション・テーブルを保持する。

図２０及び図２１は、本発明を実際の情報処理に適用する場合の、処理サーバー（アプリケーション・サーバー）とデータベース・サーバーの関係を示したものである。図２０では、ロード・バランサーを用いて、外部からの処理要求をアプリケーション・サーバーに振り分けている場合である。図２１は、処理要求振り分けシステムを作成してアプリケーション・サーバーとデータベース・サーバーを接続する場合である。

図２０の場合、外部からの処理要求は、ロード・バランサーによって、アプリケーション・サーバーに割り当てられる。従来の方法では、各アプリケーション・サーバ

ーの右側には、データベース・サーバーが一つ接続されることになるが、本発明の場合には、各アプリケーション・サーバーに対して、プライマリー・システムかアクセラレーター・システムが接続されることになる。

処理サーバー０に対する処理要求に基づいてデータの検索を行う場合には、プライマリー・システムに対して要求を出す。プライマリー・システムでは、その要求が主キーである場合にはロケーション・テーブルを使用して目的のレコードを検索する。代替キーである場合には、代替キー・ロケーション・テーブルを使用して代替キー・エントリーを検索し、その代替キー・エントリーに基づいてロケーション・テーブルを検索して、目的のレコードを検索する。

処理サーバー１に対する処理要求に基づいてデータの検索を行う場合には、アクセラレーター・システム１に対して要求を出す。アクセラレーター・システムでは、その要求が主キーである場合にはアクセラレーター・システム１のロケーション・テーブルを使用して目的のロケーション・テーブル・エントリーを探し出す。ロケーション・テーブル・エントリーには、対象ブロックのアドレスが記述してあるが、アクセラレーター・システム１にはブロックが無い。これは、プライマリー・システムのブロック及びブロック内のレコードを参照するからである。

検索が代替キーである場合には、アクセラレーター・システム１の代替キー・ロケーション・テーブルを使用して目的の代替キー・ロケーション・テーブル・エントリーを見つけ出す。代替キー・ロケーション・テーブル・エントリーには対象となる代替キー・ブロックのアドレスが格納されている。アクセラレーター・システムには、代替キー・ブロックが無いので、プライマリー・システムの代替キー・ブロックを検索する。代替キー・エントリーを検索し、その代替キー・エントリーに基づいてアクセラレーター・システム１のロケーション・テーブルを検索して、目的のロケーション・テーブル・エントリーを検索する。その次には、プライマリー・システムのブロック及びブロック内のレコードを見つけ出す。

以上が、アクセラレーター・システム１の場合に関する説明であるが、アクセラレーター・システムが複数存在する場合の、アクセラレーター・システム２以下もアクセラレーター・システム１の場合と全く同様である。

前記例では、データの検索の場合に関して記述したが、これは、レコードの追加・更新・削除の場合も全く同様である。アクセラレーター・システム１に対するデータ処理要求に基づいてアクセラレーター・システム１内で、該当ロケーション・テーブル・エントリーの検索までを行い、その後、プライマリー・システムのブロック及びブロック内のレコードの検索を行い、更に更新・追加・削除を行う。

以上のようにアクセラレーター・システムを設けて、ロケーション・テーブルと代替キー・ロケーション・テーブルの複製を用意することにより、データベースに対する負荷を分散することが可能となる。また、アクセラレーター・システムは、ブロック及び代替キー・ブロックを保持しないため、データ格納容量としては、プライマリー・システムに比較して、数十分の１から数百分の１で済み、システムを拡張する場合に必要な費用が非常に少なくて済むことになる。

図面の簡単な説明

図１は、プライマリー・システムの代表的な例である。

図２は、プライマリー・システムとアクセラレーター・システムの代表的な例である。

図３は、プライマリー・システムとアクセラレーター・システムが別のハードウエア上に存在している場合を示している。

図４は、プライマリー・システムで、レコードを追加する場合の例である。

図５は、プライマリー・システムで、レコードを追加（挿入）する場合の、ブロックの状況の例である。

図６は、プライマリー・システムで、レコードを追加（挿入）する場合の、ブロックの状況の例で、オーバーフロー・ブロックが発生する場合の例である。

図７は、プライマリー・システムで、レコードを追加（挿入）する場合の例である。レコードを追加したブロック４の主キー値の最大値が変更され、代替キーＢの代替キー・ブロック７に代替キー・オーバーフロー・ブロックが追加され、当該代替キー・

ブロックの代替キー値の最大値が変更された場合を示している。

図８は、プライマリー・システムで、レコードの追加（挿入）する場合に、当該ブロックの主キー値の最大値が変更される場合を示している。

図９は、アクセラレーター・システムで、主キー値による検索を行う場合の例である。

図１０は、アクセラレーター・システムで、代替キー値によるレコードの検索を行う場合の例である。

図１１は、アクセラレーター・システムで、レコードの追加を行う場合の例である。この場合、レコードの追加によって代替キー・エントリーの追加が行われるが、代替キーＡの代替キー・エントリーが追加された状態を示している。

図１２は、アクセラレーター・システムで、主キーによるレコードの更新を行う場合の例である。この場合、レコードの更新によって代替キー・エントリーの追加・削除が行われるが、代替キーＡとＢの代替キー・エントリーが追加・削除された状態を示している。

図１３は、アクセラレーター・システムで、代替キーによるレコードの更新を行う場合の例である。この場合、レコードの更新によって代替キー・エントリーの追加・削除が行われるが、代替キーＢとＣの代替キー・エントリーが追加・削除され、代替キー・ロケーション・テーブルＡＬＢ０の変更が発生した状態を示している。

図１４は、対称型アクセラレーター・システムの例である。

図１５は、非対称型アクセラレーター・システムの例である。

図１６は、非対称型アクセラレーター・システムに、代替キー・ロケーション・テーブルＡＬＡ２を追加する場合の例である。

図１７は、非対称型アクセラレーター・システムで、プライマリー・システムが複数のハードウエアに分散している場合の例である

図１８は、代替キー・エントリーが追加された状態の例である。

図１９は、代替キー・エントリーが追加され、代替キー・オーバーフロー・ブロックが発生した状態の例である。

図２０は、ロード・バランサーを使用した要求振り分けの例である。

図２１は、要求振り分けシステムを使用した要求振り分けの例である。

図２２は、要求振り分けシステムを使用した場合の、要求振り分けの論理の例である。

図２３は、同一のハードウエアにプライマリー・システムとアクセラレーター・システムを格納した例である。

実施例

本発明は、（日本国特許第３３４５６２８号、米国特許第６４１５３７５号、米国特許第６５８４５５５号参照）「データ格納検索方式」の発想を踏まえて、基本部分をそのまま利用しながら、スケーラビリティの確保を図ることを目的としている。

本発明では、スケーラビリティを確保する方法として、「データ格納検索方式」および「データベース再編成システム」で発明した、ロケーション・テーブルと代替キー・ロケーション・テーブルを複数用意し、それらに対するバイナリー・サーチの負荷を分散することにより全体の処理量を大幅に増加せしめるものである。

ここで、ロケーション・テーブルとブロック、代替キー・テーブル（種類ごとに１つの代替キー・ロケーション・テーブルと代替キー・ブロック）の組合せをプライマリー・システムと呼ぶ。図１が、プライマリー・システムの例である。ここでは、代替キーが３種類作成されている。

ロケーション・テーブル、代替キー・ロケーション・テーブルを保有するアクセラレーター（加速器）またはアクセラレーター・システム（加速システム）と呼ぶことにする。図２はアクセラレーター・システムの例である。アクセラレーター・システムでは、プライマリー・システムと同一のロケーション・テーブル（これを、プライマリー・システムのロケーション・テーブルと区別するために、本明細書では、フランド（ｆｒｏｎｄ）・ロケーション・テーブルと呼ぶことがある）と、代替キー・ロケーション・テーブル（これを、プライマリー・システムの代替キー・ロケーション・テーブルと区別するためにフランド代替キー・ロケーション・テーブルと呼ぶことがある）を持つ。一方で、ブロックと代替キー・ブロックを持たない。フランド・ロケーション・テーブルは、プライマリー・システムのロケーション・テーブルと同一の

エントリーを保持することが好ましいが、最低限、ブロック・アドレスまたはブロック番号を保持することが必要である。

まず、プライマリー・システム１つについて、アクセラレーター・システム１つという構成に関して述べる。

図２はプライマリー・システムとアクセラレーター・システムの構成例である。アクセラレーター・システムのフランド・ロケーション・テーブルＬ１とフランド代替キー・ロケーション・テーブルＡＬＡ１、ＡＬＢ１、ＡＬＣ１はそれぞれ、プライマリー・システムのロケーション・テーブルＬ０、代替キー・ロケーション・テーブルＡＬＡ０、ＡＬＢ０、ＡＬＣ０と同一の構成、構造、内容である。つまり、プライマリー・システムのロケーション・テーブルＬ０に対して、アクセラレーター・システムのロケーション・テーブルＬ１が割り当てられる。同様に、プライマリー・システムの代替キー・ロケーション・テーブルＡＬＡ０に対して、アクセラレーター・システムの代替キー・ロケーション・テーブルＡＬＡ１が割り当てられる。同様に、ＡＬＢ０に対して、ＡＬＢ１が割り当てられる。同様に、ＡＬＣ０に対してＡＬＣ１が割り当てられる。また、ロケーション・テーブルのどのエントリーまでが使用されているかを示す、最終ポインターと、代替キー・ロケーション・テーブルのどのエントリーまでが使用されているかを示す代替キー・最終ポインターも、プライマリー・システムとアクセラレーター・システムで保有している。「Ｆ．Ｐ.」と表示してある矢印である。これよりも後の図面では、図面を簡潔にするために、最終ポインターを省略している。

アクセラレーター・システムは、ソフトウエアのみで実現することも可能であるし、ハードウエアと組み合わせて実現することも可能である。

図２は、アクセラレーター・システムの基本部分に関して図示したものであるが、ソフト的に実現した場合の図でもある。ＣＰＵやメモリーなどはコンピューターに備わっているものを使用すればよい。プライマリー・システムとアクセラレーター・システムは、ハード的に同一であっても良いし、別のハードウエアで実現しても良い。

図３は、ハードウエアを意識した図面である。この図面では、プライマリー・システムとアクセラレーター・システムは別のハードウエア上に存在している場合を示し

ている。プライマリー・システムでは、ロケーション・テーブル、ブロック、代替キー・ロケーション・テーブル、代替キー・ブロックの他に、ＣＰＵ、通信機構、変更情報送信機構を保有している。ＣＰＵは、ロケーション・テーブルや代替キー・ロケーション・テーブルに対するバイナリー・サーチを行う他、ブロックや代替キー・ブロックの更新、ロケーション・テーブル、代替キー・ロケーション・テーブルの更新を行う。

アクセラレーター・システムでは、フランド・ロケーション・テーブル、フランド代替キー・ロケーション・テーブルの他に、ＣＰＵ、通信機構、変更情報反映機構を保有する。ＣＰＵは、フランド・ロケーション・テーブル、フランド代替キー・ロケーション・テーブルに対するバイナリー・サーチを行う。

通信機構は、プライマリー・システムからの変更情報を受け取ることと、アクセラレーター・システムで変更情報を完了した時点で、変更情報反映完了情報をプライマリー・システムに送信する。変更情報反映機構では、プライマリー・システムから送られてきた変更情報に基づき、必要な更新を行う。変更情報は、差分とすることが情報量の面から好ましいが、該当エントリーを含んだ情報であれば構わない。

この図面では、ＣＰＵは１台としてあるが、複数台用意して、ロケーション・テーブルと代替キー・ロケーション・テーブルの各々に割り当てるようにすることも可能である。また、通信機構や変更情報反映機構は、ソフトウエアのみとして、図中のＣＰＵがハードウエアとして機能することも可能であるし、各々の機構に専用のＣＰＵを設けることも可能である。

本明細書では、変更情報送信機構、変更情報反映機構、通信機構を通じて情報の送受信を行うような説明を行っているが、特別なハードウエアを作成することにより実現することも可能である。特定の番地の情報を変更する場合に、自動的に別の番地の情報を同じように書き換えるようにすることにより、プライマリー・システムのロケーション・テーブルと代替キー・ロケーション・テーブルが変更された場合に、アクセラレーター・システムのフランド・ロケーション・テーブルとフランド代替キー・ロケーション・テーブルを更新するような機構（同期更新機構）である。

プライマリー・システムからは、送信する方法だけでなく、通知する方法も可能である。

[プライマリー・システムでのアクセス方式]

以下では、主に方法に関しての説明を行うが、この方法を用いてシステム（プログラム）を作成することが可能である。

プライマリー・システムにおけるアクセスにおいて、主要な動作はアクセラレーター・システムを採用しない場合の動作と同一である。アクセラレーター・システムが存在する場合の最大の違いは、プライマリー・システムにおいて、ロケーション・テーブルと代替キー・ロケーション・テーブルの何れか、または双方に変更が在った場合に、プライマリー・システム上で変更を行うのみでなく、その変更情報をアクセラレーター・システムに送信し、アクセラレーター・システムのフランド・ロケーション・テーブルやフランド代替キー・ロケーション・テーブルを更新することにある。

「データバックアップ・リカバリー方式」では、プライマリー・システムに格納されているデータに対する変更を、セカンダリー・システムに反映させる方式として、同期密結合方式と非同期疎結合方式を提案してあるが、本発明では、この内、同期密結合方式を用いて、プライマリー・システムでの変更を、アクセラレーター・システムに反映させる説明を行う。同期密結合方式を用いることで実施が容易になるが、一般に知られている差分送信などの方法によっても実現は可能である。同期密結合方式とは、プライマリー・システムで変更があった場合に、バックアップ用のセカンダリー・システムに変更情報を送信し、セカンダリー・システムで該当する変更を完了するまで、プライマリー・システムが次の処理に入らない方式のことである。セカンダリー・システムは、「データバックアップ・リカバリー方式」で用いるバックアップ用のシステムのことで、本発明では、アクセラレーター・システムと読み替えることで適用できる。

[ロケーション・テーブルの更新をアクセラレーター・システムに反映させる方式]

まず、プライマリー・システム０におけるロケーション・テーブルＬ０の更新を、アクセラレーター・システム１に反映させる方式に関して述べる。ロケーション・テーブルＬ０が更新される、ということは、ロケーション・テーブルＬ０の或るエントリーの内容が更新される、ということである。この場合、プライマリー・システム０

のロケーション・テーブルＬ０のエントリー「ｎ」の内容が更新された場合、ロケーション・テーブルＬ０のそのエントリー番号「ｎ」と更新後の内容を、アクセラレーター・システム１に送信する。送信は、既述したように同期密結合方式を用いて行う。また、トランザクションの種類としては、Ａログ（更新後ログ）となる。また、どのファイルに対する更新かを示すために、ロケーション・テーブルＬ０の識別を付する。この識別とＡログは、「データバックアップ・リカバリー方式」で述べられているものである。

更新は、内容の更新のほかに、エントリーの追加がある。

この場合、エントリーの「ｍ」まで使用しており、「ｍ＋１」を新たに使用する、ということになる。そうすると、エントリー「ｍ＋１」の内容に関しては、更新の場合と全く同一であることになる。つまり、プライマリー・システム０のロケーション・テーブルＬ０のエントリー「ｍ＋１」の更新後の内容を、エントリーの番号と共に、アクセラレーター・システム１に、変更情報送信機構から通信機構を通じて送信する。アクセラレーター・システムでは、通信機構を通じて情報を受信し、変更情報反映機構により、該当のフランド代替キー・ロケーション・テーブル・エントリーの更新を行う。但し、この場合、最終ポインターも更新されているので、最終ポインターの内容も、エントリーと同様に、アクセラレーター・システム１に送信する必要がある。

次に、予め作成しておいたロケーション・テーブルを総て使用してしまい、全く新たにロケーション・テーブルを作成し、そのロケーション・テーブルに対してエントリーを作成する場合に関して、記述する。

この場合は、最初に作成したロケーション・テーブルのエントリーの数がｋ個だとする。ロケーション・テーブルの追加はエントリー１つ分づつ行うことも可能であるが、効率が良くないため、ある程度まとまったロケーション・テーブルを、連続領域に作成することが好ましい。そこで、「ｋ＋１」から「ｋ＋ｈ」まで、ｈ個のエントリーを持つロケーション・テーブルを追加する場合に関して記述する。

この場合、まず、連続領域にｈ個のエントリー分の領域を持つロケーション・テーブルを作成する。ｈ個のエントリーの、先頭アドレスと最終アドレスが確定する。その双方のアドレスと、ロケーション・テーブルの追加であることを示す識別を、プライマリー・システム０からアクセラレーター・システム１に、変更情報送信機構から

通信機構を通じて送信する。このトランザクションに基づき、アクセラレーター・システム１では、フランド・ロケーション・テーブルＬ１の追加作成を行う。次に、プライマリー・システム０のロケーション・テーブルＬ０の「ｋ＋１」のエントリーが更新されることになる。何故ならば、ｋ番目のエントリーに格納できなくなり、追加ロケーション・テーブルを作成したからである。

そこで、「ｋ＋１」のＡログをプライマリー・システム０からアクセラレーター・システム１へ、変更情報送信機構から通信機構を通じて送信する。このＡログの送信と、Ａログに基づくアクセラレーター・システム１の更新は、通常の更新と同様である。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

レコードの挿入に伴い、プライマリー・ブロックにオーバーフロー・ブロックを追加する場合は、次の様になる。

ロケーション・テーブルＬ０に、当該ブロックに格納されているレコードの主キー値の最大値、最小の両方を保持しない場合は全く問題ない。ロケーション・テーブルＬ０のエントリーに変更が発生しないため、プライマリー・システム０からアクセラレーター・システム１へ情報を送信する必要はない。

ロケーション・テーブルＬ０に主キー値の最小値、最大値の両方、若しくは、何れか一方を保持している場合で、最小値または最大値が変更されなければ、同様に、プライマリー・システム０からアクセラレーター・システム１へ情報を送信する必要はない。

ロケーション・テーブルＬ０に主キー値の最小値、最大値の両方、若しくは、何れか一方を保持している場合で、最小値または最大値の何れかが変更された場合には、エントリー情報の変更になるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーの更新後の内容を、エントリーの番号と共に、アクセラレーター・システム１に、変更情報送信機構から通信機構を通じて送信する。アクセラレーター・システムでは、通信機構を通じて情報を受信し、変更情報反映機構により、該当のフランド・ロケーション・テーブル・エントリーの排他を行い、その後更新を

行う。その後、排他の解除を行い、変更情報反映をプライマリー・システムに送信する。

[代替キー・テーブル・エントリーの更新をアクセラレーター・システムに反映する方式]

次に、代替キー・テーブルの更新に関して述べる。代替キー・テーブルは、代替キー・ロケーション・テーブルと代替キー・プライマリー・ブロックと代替キー・オーバーフロー・ブロックとで構成されている。代替キー・オーバーフロー・ブロックは、代替キー・プライマリー・ブロックに従属するものとして位置付けられ、代替キー・ロケーション・テーブルで管理されているのは、代替キー・プライマリー・ブロックのみである。

ここで、代替キー・エントリーの追加があった場合に関して、まず述べる。代替キー・プライマリー・ブロックには図６で示すようなエントリーが格納されている。この代替キー・プライマリー・ブロックに対して、エントリー「ｙ」がエントリー「ｘ」の直後に追加される場合は、代替キー・プライマリー・ブロックの内容は更新されるが、当該代替キー・プライマリー・ブロックの最大値、最小値の変更は無いし、代替キー・オーバーフロー・ブロックの追加も発生しないので、代替キー・ロケーション・テーブルの更新は発生しない。よって、プライマリー・システム０から、アクセラレーター・システム１に対する、更新情報の送信は無い。

次に、代替キー・プライマリー・ブロックのみの状態から、エントリーの挿入により、代替キー・オーバーフロー・ブロックが作成される場合について述べる。図６を用いて説明する。この場合、プライマリー・システムにおいて、目的の代替キー・プライマリー・ブロック「ｐ」に代替キー・エントリー「ｙ」を格納しようとしたが、代替キー・プライマリー・ブロック「ｐ」に十分な空きスペースが無く格納できない状態となる。プライマリー・システムでは、代替キー・オーバーフロー・ブロックの追加を行う必要がある。これは、まず、データ格納エリアで代替キー・オーバーフロー・ブロックを作成するスペースを探し、領域の確保を行う。代替キー・オーバーフロー・ブロックの大きさは、代替キーの種類毎に一定の大きさである。

その後、プライマリー・システム０では、新たな代替キー・エントリー「ｙ」を代

替キー・プライマリー・ブロックの代替キー・エントリー「ｚ」があった場所に格納を行い、それに伴いエントリー「ｚ」が代替キー・オーバーフロー・ブロックに格納される。代替キー・プライマリー・ブロックの代替キー値の最小値と最大値の内、最大値が「ｙ」になったため、最大値の情報を「ｙ」に書き換える。また、代替キー・オーバーフロー・ブロックの代替キー値の最小値、最大値を「ｚ」とする。

この場合も、代替キー・ロケーション・テーブルのエントリーの変更は無いため、プライマリー・システム０から、アクセラレーター・システム１に対する、更新情報の送信は無い。

次に、代替キー・ロケーション・テーブルのエントリーが更新される場合に関して述べる。図８に示すように、更新前の代替キー・プライマリー・ブロックの最後には、「ｘ」と「ｙ」の代替キー・エントリーが格納されている。データの更新・追加によって、新たに代替キー・エントリー「ｚ」が追加されると、この代替キー・プライマリー・ブロックの「ｙ」の後に、代替キー・エントリー「ｚ」が追加されることになる。

この場合、代替キー・プライマリー・ブロック「ｐ」の代替キー値の最大値が、「ｙ」から「ｚ」に更新されたことになる。この場合は、プライマリー・システム０から、アクセラレーター・システム１に対して、Ａログとして、代替キー・エントリーの更新後の値と、どの代替キー・ブロックに対する更新であるかの情報、この場合は代替キー・ブロック番号「ｐ」を送信する。この送信は同期密結合方式を使用して、変更情報送信機構から通信機構を通じて送信される。

アクセラレーター・システム１では、Ａログを受信したら、変更情報反映機構が、直ちに代替キー・ロケーション・テーブルの当該エントリーを排他し、当該代替キー・エントリーの更新を行った後、排他を解除し、更新完了をプライマリー・システム０に通知する。

[主キーによる検索：プライマリー・システムの場合]

プライマリー・システム０における主キーを用いた検索は、「データ格納検索方式」で述べてあるとおりである。

検索は、プライマリー・システム０のロケーション・テーブルＬ０に対して、Ｌ０

の先頭アドレスと、Ｌ０の最終ポインターが指しているロケーション・テーブル・エントリーの間に対してバイナリー・サーチを行い、ターゲット・キー値が含まれるロケーション・テーブル・エントリーを検索する。以下の説明では、説明を簡潔にするために、単に、ロケーション・テーブルに対してバイナリー・サーチを行う、と記述する場合があるが、実際には、ロケーション・テーブルの先頭アドレスと、ロケーション・テーブルの最終ポインターが指しているロケーション・テーブル・エントリーの間に対してバイナリー・サーチを行うことを意味している。

ロケーション・テーブル・エントリーに、そのエントリーが管理しているブロックの主キー値の最小値と最大値の両方若しくは一方を保持している場合には、その最小値と最大値とターゲット・キー値の比較を行う。主キー値の最小値と最大値をロケーション・テーブル・エントリーが保持していない場合には、ブロック内の主キー値の最小値と最大値の双方または一方と比較する。最小値と最大値の一方のみ保有している場合には、その前後のロケーション・テーブル・エントリーまたはブロックと比較する必要がある。このようにして、ターゲット・キー値を持つレコードが含まれるブロックを検索する。ブロック内には、そのターゲット・キー値を持つレコードが存在する場合と、存在しない場合がある。

検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

[代替キーによる検索：プライマリー・システムの場合]

プライマリー・システム０における代替キーを用いた検索は、「データ格納検索方式」で述べてあるとおりである。

ターゲット・キー値（検索の目的となるキーの値）によって検索を行う。この場合、代替キーＡ０に対する検索であるとする。検索は、プライマリー・システム０の代替キー・ロケーション・テーブルＡＬＡ０に対して、ＡＬＡ０の先頭アドレスと、ＡＬＡ０の代替キー・最終ポインターが指している代替キー・ロケーション・テーブル・エントリーの間に対して、バイナリー・サーチを行い、ターゲット・キー値が含まれる代替キー・ロケーション・テーブル・エントリーを検索する。以下の説明では、説

明を簡潔にするために、単に、代替キー・ロケーション・テーブルに対してバイナリー・サーチを行う、と記述する場合があるが、実際には、代替キー・ロケーション・テーブルの先頭アドレスと、代替キー・ロケーション・テーブルの代替キー・最終ポインターが指している代替キー・ロケーション・テーブル・エントリーの間に対してバイナリー・サーチを行うことを意味している。

次に、その代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロック内を検索し、目的の代替キー・エントリーを見つけ出す。その代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合は、代替キー・オーバーフロー・ブロックも検索の対象とする。

検索した代替キー・エントリーが、代替キー・ブロック番号「ｎ」を指していたとする。そこで、プライマリー・システム０上の代替キー・ブロック「ｎ」にターゲット・キー値を持つ代替キー・エントリーが存在すれば、その代替キー・エントリーが目的のエントリーとなる。ブロック「ｎ」にターゲット・キー値を持つ代替キー・エントリーが存在しなければ、そのターゲット・キー値を持つレコードが存在しないということになる。

目的のレコードを検索するためには、ロケーション・テーブルＬ０において、ターゲット・レコードを含むブロックを探す必要がある。

代替キー・エントリーにブロック番号を保持している場合は、ロケーション・テーブルＬ０のエントリーの長さと、ブロック番号を掛けたものが、ロケーション・テーブルＬ０の先頭からの変位となる。そうして検索したロケーション・テーブル・エントリーが指しているブロックに目的のレコードが格納されていることになる。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

代替キー・エントリーにブロック・アドレスを保持している場合は、代替キー・エントリーが指しているブロック・アドレスのブロックに目的のレコードが格納されていることになる。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。この場合には、レコードが見つからないという状態は発生しない。

ブロック番号及びブロック・アドレスの何れも保持していない形式である場合には、

検索で求めた代替キー・エントリーが挿している主キー値を持つレコードの検索を行うため、プライマリー・システムのロケーション・テーブルＬ０上で、当該主キー値をターゲット・キー値としてバイナリー・サーチを行い、ロケーション・テーブル・エントリーを検索する。そこで求めたブロック番号を元に、プライマリー・システムのブロック内で対象レコードの検索を行う。検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

代替キーは、ノン・ユニークなキーであるので、同一の代替キー値を持つレコードが複数存在する可能性があるがその場合は、上記の操作を繰り返す。

[レコードの追加、更新、削除：プライマリー・システムの場合]

以上で主キーおよび代替キーでの検索の場合を述べた。検索で用いた方式を応用することで、レコードの追加、更新、削除が行える。

[レコードの追加：プライマリー・システムの場合]

まず、レコードの追加の場合に関して図４を参照しながら説明する。
レコードを追加する場合には、そのレコードがどのブロックに格納されるかを見つける必要がある。

プライマリー・システム０上で行う場合には、まず、ロケーション・テーブルＬ０をバイナリー・サーチして、追加するレコードが格納されるべきロケーション・テーブル・エントリーを検索する。

検索したエントリーが、ブロック番号「ｎ」を指していたとする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。排他は、「データベース再編成システム、並びに、データベース」で発明した、ロケーション・テーブル・エントリーとブロックに排他情報を書き込む方式が高速化のためには好まし

い。

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値より小さい主キー値の内、最大の主キー値を持つレコードを見つける。そのレコードの次のアドレスに、当該レコードを挿入することになる。レコードを挿入するためには、挿入するレコードより、上位アドレスにあるレコードを挿入レコードが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、オーバーフロー・ブロックが発生する場合には、オーバーフロー・ブロックの追加を行い、レコードの移動を行う。その後、レコードの追加を行う。図４のブロック番号４の中の黒い長方形が、追加されたレコードを示している。この場合にはブロック番号４のブロックに格納されている。図５は、より具体的に、レコードの挿入の状況を図示している。この場合は、レコードｘとレコードｚの間に、レコードｙが挿入される例を示している。図６では、レコードの挿入によって、オーバーフロー・ブロックが発生する場合を示している。この場合も、図５と同様に、当該ブロック内の主キー値の最大値は変更されない。

レコードが挿入となる場合には、ロケーション・テーブルＬ０の変更は行われない。レコードの追加によって、ロケーション・テーブルが変更される場合は、ロケーション・テーブル・エントリーに主キー値の最大値を持ち、追加されるレコードの主キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、ロケーション・テーブルの当該ロケーション・テーブル・エントリーの情報の更新を行う。それに伴って、プライマリー・システムからアクセラレーター・システム１に対して当該ロケーション・テーブル・エントリーの更新情報を、変更情報送信機構から通信機構を通じて送信する。アクセラレーター・システムでは、通信機構を通じて受信し、変更情報反映機構により、当該フランド・ロケーション・テーブル・エントリーの更新を行う。図４では、ロケーション・テーブル・エントリーの変更が無かった場合を示している。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

[レコードの追加に伴う代替キー・エントリーの追加：プライマリー・システムの場合]

レコードが追加されると、それに伴って代替キー・エントリーの追加が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。

代替キーが３種類ある場合には、レコード追加によって３種類の代替キー・エントリーの追加が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

代替キー・エントリーがＡｅ、Ｂｅ、Ｃｅの３つ作成される。この各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡ０の場合に関して述べる。代替キー・ロケーション・テーブルＡＬＡ０のｅの代替キー値をターゲット・キー値としてバイナリー・サーチする。それによって、当該代替キー・エントリーが格納されるべき代替キー・ブロックを検索する。

この排他順序は、「データベース再編成システム」で述べてあるように、ロケーション・テーブル、ブロック、代替キー・ロケーション・テーブル、代替キー・ブロックの排他順序が一定であるので、デッドロックが発生する可能性は、非常に少なくなる。これは、これ以降の説明全般に適用できることである。

ここで、当該代替キー・ブロックにＡｅの挿入を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つエントリーが存在する場合がある。

まず、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在する場合に関して述べる。この場合は、同一の代替キー値を持つ代替キー・エントリーの内、追加対象代替キー・エントリーの主キーに着目し、追加対象代替キー・エントリーの主キー値より小さい主キー値の内、最大の主キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入する。挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく。また、上位方向に移動することによって、代替キー・オ

ーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。

次に、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在しない場合に関して述べる。この場合は、対象ブロック内の代替キー値を持つ代替キー・エントリーの内、追加対象代替キー・エントリーの代替キー値より小さい主キー値の内、最大の代替キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。代替キー・エントリーの追加により、代替キー・オーバーフロー・ブロックが発生するが、最小値や最大値に影響が無い場合の例が、図１９である。

この代替キー・エントリーの追加によって、代替キー・ロケーション・テーブルが変更される場合は、代替キー・ロケーション・テーブル・エントリーに代替キー値の最大値を持ち、追加される代替キー値が当該代替キー・ブロック中で最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。代替キー・エントリーの追加により、代替キー・ブロックの最大値が変更される場合の例が、図１８である。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を、変更情報送信機構から通信機構を通じて送信する。アクセラレーター・システムでは、通信機構を通じて受信し、変更情報反映機構が、該当のフランド代替キー・ロケーション・テーブル・エントリーの排他を行い、変更を行った後、排他の解除を行う。その後、プライマリー・システムに対して、変更情報反映完了情報を送信する。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に

実施する。図４では、Ａｅが代替キー・ブロックのブロック番号７に、Ｂｅが代替キー・ブロック番号３に、Ｃｅが代替キー・ブロック番号９に、それぞれ格納された状態を示している。この場合は、代替キー・ロケーション・テーブルの変更が無かった場合を示している。

次に、図７では、レコードの挿入によって、ロケーション・テーブルが変更され、代替キー・ロケーション・テーブルＡＬＢ０も変更される場合を示している。ここでは、レコードの追加により、当該ブロック中のレコードの主キー値の最大値が変更されたことによる。図８は、レコードの挿入を具体的に図示している。この場合、レコードｘとｙの後に、レコードｘが追加されており、当該ブロック内のレコードの主キー値の最大値が変更されている。これにより、プライマリー・システムのＬ０のロケーション・テーブル・エントリー４を変更する。この変更情報を、アクセラレーター・システムのロケーション・テーブルＬ１に対して送信する。アクセラレーター・システム１では、変更情報により、ロケーション・テーブルＬ１のロケーション・テーブル・エントリー４を変更する。代替キー・ロケーション・テーブルＡＬＡ０は変更が無かった。代替キー・ロケーション・テーブルＡＬＢ０は、Ｂｅの追加により、代替キー・ロケーション・テーブル・エントリー７が変更されることになった。プライマリー・システムでは、ＡＬＢ０のエントリー７を変更し、その変更情報をアクセラレーター・システム１に、変更情報送信機構から通信機構を通じて送信する。アクセラレーター・システム１では、通信機構を通じて受信し、変更情報反映機構が、ＡＬＢ０のエントリー７の変更を行う。代替キー・ロケーション・テーブルＡＬＣ０には変更が無かった。アクセラレーター・システムでロケーション・テーブルや代替キー・ロケーション・テーブルの変更を行う場合には、変更前に排他を行う。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加が完了したら、これまでの排他を解除する。

[主キーによるレコードの更新：プライマリー・システムの場合]

更新は、主キーによる検索と同様な方法により、目的のレコードを見つけ、そのレコードに対して更新することになる。更新の場合は、ブロックに対する排他を行う。また、主キー値が更新される場合は、レコードの更新では扱えず、レコードの削除と

追加となる。これは、従来のデータベースで行われている方法である。

ロケーション・テーブルＬ０をバイナリー・サーチして、更新するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。そのレコードを更新処理することになる。また、ブロック内にターゲット・キー値を持つレコードが存在しない場合には、レコードが存在しないということになる。

このようにして、レコードの更新が行われるが、主キー値の更新は許されないので、ロケーション・テーブルＬ０の変更は行われない。主キー値の更新を行いたい場合は、当該主キー値を持ったレコードを一旦削除し、新たな主キー値を持ったレコードを追加すると言う方法を採る。

[主キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：プライマリー・システムの場合]

レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新された場合に、代替キー・エントリーの更新が必要となる。

代替キーが３種類ある場合には、レコード更新によって０種類から３種類の代替キー・エントリーの更新が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

ここでは、すべての代替キー・エントリーが更新される場合を例にとって説明する。新しい代替キー・エントリーがＡｅｎ、Ｂｅｎ、Ｃｅｎの３つ作成され、これらの代替キー・エントリーの追加が必要になる。逆に、当該レコードのそれまでの代替キー・

エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏは不要となり、削除が必要になる。

Ａｅｎ、Ｂｅｎ、Ｃｅｎの３つの代替キー・エントリーの各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡ０の場合に関して述べる。代替キー・ロケーション・テーブルＡＬ０をＡｅｎの代替キー値をターゲット・キー値としてバイナリー・サーチする。

これ以下の動作は、レコード追加による代替キー・エントリーの追加の場合と同様であるので、省略する。また、Ｂｅｎ、Ｃｅｎに関しては、Ａｅｎの場合と同様であるので、これも省略する。

次に、代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏの各々を、代替キー・テーブルＡ，Ｂ，Ｃから削除する必要がある。前記の説明と同様に、Ａｅｏの場合に関して説明を行う。やはり、代替キー・ロケーション・テーブルＡＬ０を使用して、Ａｅｏの中にある代替キー値をターゲット・キー値として、バイナリー・サーチを行う。

ここで、Ａｅｏの削除を行うのであるが、代替キーはノン・ユニークであるので、同一の代替キー値を持つ代替キー・エントリーが複数存在する場合があるので、代替キー値と主キー値が一致する代替キー・エントリーを見つける。

次に、ターゲット・キー値と同一の代替キー値と主キー値を持つ代替キー・エントリーの削除を行う。代替キー・エントリーを削除すると、削除した代替キー・エントリーが占めていた領域が空くことになる。削除した代替キー・エントリーより、上位アドレスにある代替キー・エントリーを削除対象代替キー・エントリーが占めていた領域の分だけアドレスの下位方向に動かす。

この代替キー・エントリーの追加と削除によって、代替キー・ロケーション・テーブルが変更される場合は、代替キー・ロケーション・テーブル・エントリーが代替キー値の最大値を持ち、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合と、代替キー・ロケーション・テーブル・エントリーが代替キー値の最小値を持ち、削除される代替キー値が当該ブロック中で最小のキー値を持つ場合と、代替キー・ロケーション・テーブル・エントリーが代替キー値の最大値を持ち、削除される代替キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。

この代替キー・エントリーの追加と削除は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

[代替キーによるレコードの更新：プライマリー・システムの場合]

更新は、代替キーによる検索と同様な方法により、目的のレコードを見つけ、そのレコードに対して更新することになる。代替キーはノン・ユニークであるので、同一の代替キー値を持つレコードが複数存在する可能性があり、その場合は、同一キー値のレコードに対して更新を行う。更新の場合は、ブロックに対する排他を行う。また、代替キー値が更新される場合は、当該代替キー・テーブルに対する変更となり、旧代替キー・エントリーの削除と、新代替キー・エントリーの追加となる。

代替キー・エントリーの追加、削除によって、代替キー・ロケーション・テーブル・エントリーに変更がある場合には、プライマリー・システムからアクセラレーター・システムに対して、変更情報を送信する。

[主キーによるレコードの削除に伴う代替キー・エントリーの削除：プライマリー・システムの場合]

レコードが削除されると、それに伴って代替キー・エントリーの削除が必要になる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード削除の場合は、必ず代替キー・エントリーの更新が行われる。

代替キーが３種類ある場合には、レコード削除によって３種類の代替キー・エントリーの削除が行われる。

これまでの代替キー・エントリーＡｅ、Ｂｅ、Ｃｅの３つが削除対象となる。代替キー・ロケーション・テーブル・エントリーに変更が在る場合には、プライマリー・システムからアクセラレーター・システムに変更情報を送信する。

[代替キーによるレコードの削除：プライマリー・システムの場合]

代替キー・ロケーション・テーブルＡＬＯをバイナリー・サーチして、削除するレコードの代替キー値を持つ代替キー・ロケーション・テーブル・エントリーを検索する。

代替キー・ロケーション・テーブル・エントリーを検索した後、代替キー・ブロック内を検索し、代替キー・エントリーを見つけ出す。代替キー・エントリーが検索できたら、レコードが格納されているブロックを検索する。

この動作は、代替キーによるレコードの検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。削除を代替キーで実行する場合には、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、削除を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次削除するものとする。

レコードの削除に伴い発生する、ブロック内の空きスペースの管理や、代替キー・エントリーの削除は、主キーによるレコード削除と同様である。ロケーション・テーブル・エントリーと代替キー・ロケーション・テーブル・エントリーのうち、変更があった分について、プライマリー・システムからアクセラレーター・システムへ、変更情報を送信する。

[アクセラレーター・システムを利用したアクセスに関して]

次に、アクセラレーター・システムでの、データの検索、追加、更新、削除に伴うアクセラレーター・システム内での動作と、それに伴うプライマリー・システム内での動作、更にプライマリー・システムでの動作に伴う、アクセラレーター・システムの動作に関して説明を行う。基本的には、プライマリー・システムでの動作と似ている。しかしながら、アクセラレーター・システムには、ブロックと代替キー・ブロックが存在しないので、それらに対するアクセスは、プライマリー・システムに対して行うことになり、アクセラレーター・システムとプライマリー・システムの双方が連携して動作することが最大の違いである。プライマリー・システムで、ロケーション・

テーブルや代替キー・ロケーション・テーブルに変更が生じた場合に、プライマリー・システムからアクセラレーター・システムに変更情報を送信し、アクセラレーター・システムで当該ロケーション・テーブル、代替キー・ロケーション・テーブルの更新を行うことは、プライマリー・システムでのアクセスと同様である。

[主キーによる検索：アクセラレーター・システム１の場合]

アクセラレーター・システム１における主キーを用いた検索の場合を、図９を用いて説明する。ターゲット・キー値によって検索を行う。検索は、アクセラレーター・システム１のロケーション・テーブルＬ１を用いて、Ｌ１の先頭アドレスと、Ｌ１の最終ポインターが指しているロケーション・テーブル・エントリーの間に対してバイナリー・サーチを行い、ターゲット・キー値が含まれるロケーション・テーブル・エントリーを検索する。

検索したエントリーが、ブロック番号「４」を指していたとする。ブロックはアクセラレーター・システム１には存在せず、プライマリー・システム０にのみ存在する。

そこで、ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「４」に対して行うことになる。ブロック「４」にターゲット・キー値を持つレコードが存在すれば、そのレコードがターゲット・レコードとなる。ブロック「４」にターゲット・キー値を持つレコードが存在しなければ、そのレコードが存在しないということになる。この場合は、ブロック番号４であるが、一般的にブロック番号「ｎ」とすることができる。これは、以下の説明でも同様である。

検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

[代替キーによる検索：アクセラレーター・システム１の場合]

アクセラレーター・システム１における代替キーを用いた検索の場合を、図１０を用いて説明する。ターゲット・キー値によって検索を行う。この場合は、代替キーＡに対する検索であるとする。検索は、アクセラレーター・システム１の代替キー・ロケーション・テーブルＡＬＡ０に対して、バイナリー・サーチを行い、ターゲット・

キー値が含まれる代替キー・ロケーション・テーブル・エントリーを検索する（図１０の８角形の１）。

検索した代替キー・エントリーが、代替キー・ブロック番号「８」を指していたとする。代替キー・ブロックはアクセラレーター・システム１には存在せず、プライマリー・システム０にのみ存在する。

そこで、代替キー・ブロックの代替キー・エントリーに対するアクセスは、プライマリー・システム０にある代替キー・ブロック「８」に対して行う。代替キー・ブロック「８」にターゲット・キー値を持つ代替キー・エントリーが存在すれば、その代替キー・エントリーが目的のエントリーとなる。代替キー・ブロック「８」にターゲット・キー値を持つ代替キー・エントリーが存在しなければ、そのターゲット・キー値を持つレコードが存在しないということになる。代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、代替キー・オーバーフロー・ブロックも検索の対象となる（図１０の８角形の２）。ここでは、代替キー・ブロック番号８の場合を例にとったが、一般的には代替キー・ブロック番号「ｎ」と表現できる。これは、以下の説明でも同様である。

これまでで、代替キー・エントリーの検索を終了した。代替キー・エントリーには、代替キー値とその代替キー値を持つレコードの主キー値の他、必要に応じて、レコードが格納されているブロックの番号、またはブロックのアドレスを保持している。

目的のレコードを検索するためには、再度、アクセラレーター・システム１で、ロケーション・テーブルＬ１を使用して、ターゲット・レコードを含むブロックを探す必要がある。

代替キー・エントリーにブロック番号を保持している場合は、アクセラレーター・システム１のロケーション・テーブルＬ１を直接的に検索する。ロケーション・テーブルＬ１のエントリーの長さと、ブロック番号を掛けたものが、ロケーション・テーブルＬ１の先頭からの変位となるからである。そうして検索したロケーション・テーブル・エントリーが指しているブロックに目的のレコードが格納されていることになる。ブロックはプライマリー・システム０のブロックを参照する。当該ブロック内で、検索で求めた代替キー・エントリーが指している主キー値を持つレコードの検索を行う。

代替キー・エントリーにブロック・アドレスを保持している場合は、プライマリー・システム０の当該ブロックを参照する。当該ブロック内で、当該代替キー・エントリーが指している主キー値を持つレコードを見つける。

ブロック番号及びブロック・アドレスの何れも保持していない形式である場合には、アクセラレーター・システム１のロケーション・テーブルＬ１上で、当該主キー値をターゲット・キー値としてバイナリー・サーチを行い、ロケーション・テーブル・エントリーを検索する（図１０の８角形の３）。そこで求めたブロック・アドレスを元に、プライマリー・システム０の当該ブロック内で対象レコードの検索を行う（図１０の８角形の４）。検索の場合は、レコードの更新が発生しないため、ロケーション・テーブルの更新も発生せず、プライマリー・システムからアクセラレーター・システムに対して情報の送信を行うことは無い。また、原則的に排他は不要である。

[レコードの追加、更新、削除：アクセラレーター・システムの場合]

以上で主キーおよび代替キーでの検索の場合を述べた。検索で用いた方式を応用することで、レコードの追加、更新、削除が行える。

レコードの追加は主キーによって実行される。

まず、レコードの追加の場合に関して、図１１を用いて説明する。

[レコードの追加：アクセラレーター・システムの場合]

アクセラレーター・システム１上でレコードの追加を行う場合には、まず、ロケーション・テーブルＬ１をバイナリー・サーチして、追加するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する（図１１の８角形の１）。

検索したエントリーが、ブロック番号「８」を指していたとする。ブロックはアクセラレーター・システムには存在せず、ブロックプライマリー・システム０に存在しているので、それに対してアクセスする（図１１の８角形の２）。

この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行う。

ブロック「８」内のレコードを調べ、ターゲット・キー値より小さい主キー値の内、最大の主キー値を持つレコードを見つける。そのレコードの次のアドレスに、当該レコードを挿入することになる。挿入に伴うブロック内の移動などは、プライマリー・システムの場合と同様である（図１１の８角形の３）。

このレコードの追加によって、ロケーション・テーブルが変更される場合は、追加されるレコードの主キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、プライマリー・システムのロケーション・テーブルＬ０の当該ロケーション・テーブル・エントリーの情報の更新を行う。それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該ロケーション・テーブル・エントリーの更新情報を送信する。図１１では、ロケーション・テーブルの変更が無かった場合を図示している。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

レコード追加の特殊な場合として、最終ブロックの次に新たにブロックを作成して、そこにレコードを格納する場合があるが、新たにブロックを追加し、そのブロックを管理するためにロケーション・テーブルに新たにエントリーを追加する必要があるが、それ以降は、上述の追加の場合と同様である。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

[レコードの追加に伴う代替キー・エントリーの追加：アクセラレーター・システムの場合]

レコードが追加されると、それに伴って代替キー・エントリーの追加が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。

代替キーが３種類ある場合には、レコード追加によって３種類の代替キー・エントリーの追加が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけで

ある。よって、ここでは非登録の場合の説明を省略する。

代替キー・エントリーがＡｅ、Ｂｅ、Ｃｅの３つ作成される。この各々を代替キー・テーブルＡ，Ｂ，Ｃに格納する必要がある。代替キー・テーブルＡの場合に関して述べる。アクセラレーター・システム１上で、代替キー・ロケーション・テーブルＡＬＡ１をＡｅの代替キー値をターゲット・キー値としてバイナリー・サーチする（図１１の８角形の４）。

目的の代替キー・ロケーション・テーブル・エントリーの検索をアクセラレーター・システムで完了したら、その代替キー・ロケーション・テーブル・エントリーに対応する、プライマリー・システム０の当該代替キー・ロケーション・テーブル・エントリーと、当該代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロックの排他を実行する（図１１の８角形の５）。つまり、代替キー・ブロックはプライマリー・システム上にのみ存在しているので、代替キー・ブロックに対する操作はプライマリー・システム上で実行する。

ここで、Ａｅの挿入を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つエントリーが存在する場合がある。

まず、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在する場合に関して述べる。この場合は、同一の代替キー値を持つ代替キー・エントリーの内、追加対象代替キー・エントリーの主キーに着目し、追加対象代替キー・エントリーの主キー値より小さい主キー値の内、最大の主キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う。

次に、ターゲット・キー値と同一の代替キー値を持つ代替キー・エントリーが存在しない場合に関して述べる。この場合は、対象ブロック内の代替キー値を持つ代替キ

ー・エントリーの内、追加対象代替キー・エントリーの代替キー値より小さい主キー値の内、最大の代替キー値を持つ代替キー・エントリーの次のアドレスに、当該代替キー・エントリーを挿入することになる。代替キー・エントリーを挿入するためには、挿入する代替キー・エントリーより、上位アドレスにある代替キー・エントリーを追加対象代替キー・エントリーが占める領域の分だけアドレスの上位方向に動かしておく必要がある。また、上位方向に移動することによって、代替キー・オーバーフロー・ブロックが発生する場合には、代替キー・オーバーフロー・ブロックの追加を行い、代替キー・エントリーの移動を行う。その後、代替キー・エントリーの追加を行う（図１１の８角形の６）。

この代替キー・エントリーの追加によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルＬ０の当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。

この送信は同期密結合方式であるので、アクセラレーター・システム１で更新が完了するまで、プライマリー・システムでは排他解除を行わず、新たなトランザクション処理で、同一のブロックなどに対する排他が発生した場合には、排他待ちになる。

この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。図１１では、代替キー・テーブルＡへの代替キー・エントリーの追加が終了した時点を示している。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加が完了したら、これまでの排他を解除する。

［主キーによるレコードの更新：アクセラレーター・システムの場合］

［主キーによるレコードの更新に伴う、ブロックの更新：アクセラレーター・システムの場合］

アクセラレーター・システム１上でレコード更新要求に基づく処理を行う場合について図１２を用いて説明する。まず、ロケーション・テーブルＬ１をバイナリー・サ

ーチして、更新するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する（図１２の８角形の１）。

検索したエントリーが、ブロック番号「４」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「４」に対して行うことになる（図１２の８角形の２）。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行う。

ブロック「４」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つけ、そのレコードを更新処理することになる（図１２の８角形の３）。また、ブロック内にターゲット・キー値を持つレコードが存在しない場合には、レコードが存在しないということになる。図１１では、レコードが存在する場合を示している。ここでは、ブロック番号４の場合に関して述べたが、一般的には、ブロック番号「ｎ」とすることができる。

このようにして、レコードの更新が行われるが、主キー値の更新は許されないので、ロケーション・テーブルＬ０の変更は行われない。主キー値の更新を行いたい場合は、当該主キー値を持ったレコードを一旦削除し、新たな主キー値を持ったレコードを追加すると言う方法を採る。よって、主キー値が「変更」される場合は、レコード追加と削除の説明で十分である。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

[主キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：アクセラレーター・システムの場合]

レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新され

た場合に、代替キー・エントリーの更新が必要となる。

代替キーが３種類ある場合には、レコード更新によって０種類から３種類の代替キー・エントリーの更新が行われる。非登録キーのように代替キー・エントリーを作成しない場合は、代替キー・エントリーの追加が無いだけなので、動作としては単純になるだけである。よって、ここでは省略する。

ここでは、すべての代替キー・エントリーが更新される場合を例にとって説明する。新しい代替キー・エントリーがＡｅｎ、Ｂｅｎ、Ｃｅｎの３つ作成され、これらの代替キー・エントリーの追加が必要になる。逆に、当該レコードのそれまでの代替キー・エントリーＡｅｏ、Ｂｅｏ、Ｃｅｏは不要となり、削除が必要になる。図１２では、代替キー・テーブルＡとＢに関しては変更され、Ｃに関しては変更されない場合を示している。

Ａｅｎ、Ｂｅｎの２つの代替キー・エントリーの各々を代替キー・テーブルＡ，Ｂに格納する必要がある。代替キー・テーブルＡ０の場合に関して述べる。アクセラレーター・システム上で、代替キー・ロケーション・テーブルＡＬ１をＡｅｎの代替キー値をターゲット・キー値としてバイナリー・サーチする（図１２の８角形の４）。

これ以下の動作は、レコード追加による代替キー・エントリーの追加の場合と同様であるので、省略する（図１２の８角形の５，６）。また、Ｂｅｎに関しては、Ａｅｎの場合と同様である。図１２では、Ｂｅｎの追加に伴い、代替キー・ロケーション・テーブル・エントリーＡＬＢ０が変更になる場合を示している。この場合は、図１２の８角形の７、８、９の動作後に、代替キー・ロケーション・テーブル・エントリーの変更情報を、アクセラレーター・システム１に送信する。アクセラレーター・システム１では、その情報で、当該代替キー・ロケーション・テーブル・エントリーを更新する（図１２の８角形の１０）。

次に、代替キー・エントリーＡｅｏ、Ｂｅｏの各々を、代替キー・テーブルＡ，Ｂから削除する必要がある。前記の説明と同様に、Ａｅｏの場合に関して説明を行う。やはり、アクセラレーター・システム１上で、代替キー・ロケーション・テーブルＡＬ１を使用して、Ａｅｏの中にある代替キー値をターゲット・キー値として、バイナリー・サーチを行う（図１２の８角形の１１）。

目的の代替キー・ロケーション・テーブル・エントリーの検索を完了したら、プラ

イマリー・システム０の当該代替キー・ロケーション・テーブル・エントリーと、当該代替キー・ロケーション・テーブル・エントリーが指している代替キー・ブロックの排他を実行する（図１２の８角形の１２）。

ここで、Ａｅｏの削除を行うのであるが、対象代替キー・ブロックの中の代替キー・エントリーを調べる。代替キーはノン・ユニークであるので、同一の代替キー値を持つ代替キー・エントリーが存在する場合があるので、代替キー値と主キー値が一致する代替キー・エントリーを見つけ、その代替キー・エントリーの削除を行う（図１２の８角形の１３）。代替キー・エントリーを削除した場合の、代替キー・ブロック内の動作は、プライマリー・システムでのアクセスの場合と同様である。

この代替キー・エントリーの追加は、代替キー・テーブルＢに関しても同様に実施する。図１２では、Ｂｅｏを削除する場合も示している（図１２の８角形の１４、１５、１６）。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

[代替キーによるレコードの更新：アクセラレーター・システムの場合]

図１３を使用して説明する。アクセラレーター・システム１上で行う場合には、まず、代替キー・ロケーション・テーブルＡＬＡ１をバイナリー・サーチして、更新するレコードの代替キー値を持つ代替キー・ロケーション・テーブル・エントリーを見つけ出す（図１３の８角形の１）。

代替キー・ロケーション・テーブル・エントリーを検索した後、その代替キー・ロケーション・テーブル・エントリーが指しているプライマリー・システム０上の代替キー・ブロック内を検索し、代替キー・エントリーを検索する（図１３の８角形の２）。代替キー・エントリーが検索できたら、再度、アクセラレーター・システム１上で、ロケーション・テーブルＬ１を使用して、バイナリー・サーチを行い（図１３の８角形の３）、レコードが格納されているブロックを検索する（図１３の８角形の４）。

この動作は、代替キーによる検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。更新を代替キーで実行する場合には、

同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、更新を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次更新するものとする。

検索したロケーション・テーブル・エントリーが、ブロック番号「n」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にあるブロック「n」に対して行う。

ブロック「n」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つけ（図１３の８角形の５）、レコードの更新を行う。主キー値の更新が許されないのは、主キーによるレコード更新で述べたことと同じである。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

[代替キーによるレコードの更新に伴う代替キー・エントリーの追加、削除：アクセラレーター・システムの場合]

レコードが更新されると、それに伴って代替キー・エントリーの更新、実際には追加と削除が行われる場合がある。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード更新の場合は、必ず代替キー・エントリーの更新が行われるわけではなく、レコードの項目（フィールド）で代替キーに指定した項目が更新された場合に、代替キー・エントリーの更新が必要となる。図１３では、代替キーＡの変更はなく、代替キーB と C が変更された場合を示している。この場合、代替キーB の変更によって、代替キー・ロケーション・テーブルＢ０が変更になり、その変更をアクセラレーター・システム１に送信している。

この代替キー・エントリーの追加・削除は、「主キーによるレコード更新による、代替キー・エントリーの更新」と方式が同様であるので、説明は省略する。

[主キーによるレコードの削除：アクセラレーター・システムの場合]

アクセラレーター・システム１上で行う場合には、まず、ロケーション・テーブルＬ１をバイナリー・サーチして、削除するレコードの主キー値を持つロケーション・テーブル・エントリーを検索する。

検索したエントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。

ブロック内のレコードに対する検索は、プライマリー・システム０にある、ブロック「ｎ」に対して行うことになる。この場合は、更新系の処理要求となるので、プライマリー・システム０のロケーション・テーブルＬ０の当該エントリーと、当該エントリーが指しているブロックの排他を行ってから、以下の動作を行う。

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。そのレコードを削除処理することになる。ブロック内にターゲット・キー値を持つレコードが存在しない場合は、当該レコードは存在しないことになる。

このようにして、レコードの削除が行われる。レコードの削除に伴う、ブロックやオーバーフロー・ブロック内の動作は、プライマリー・システムでのアクセスの場合と同様である。

このレコードの削除によって、ロケーション・テーブルが変更される場合は、削除される主キー値が当該ブロック中で最小または最大のキー値を持つ場合で且つ、代替キー・エントリーに主キー値の最小値または最大値を持っている場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

排他解除は、これから記述する代替キー・エントリーの追加が完了してから実施する。

[主キーによるレコードの削除に伴う代替キー・エントリーの削除：プライマリー・システムの場合]

レコードが削除されると、それに伴って代替キー・エントリーの削除が行われる。プライマリー・システムで代替キーを使用していない場合は不要であるが、ここでは、代替キーが３種類（Ａ，Ｂ，Ｃ）存在する場合に関して述べる。レコード削除の場合

は、必ず代替キー・エントリーの更新が行われる。
代替キーが３種類ある場合には、レコード削除によって３種類の代替キー・エントリーの削除が行われる。

これまでの代替キー・エントリーＡｅ、Ｂｅ、Ｃｅの３つが削除対象となる。
代替キー・エントリーの削除は、代替キー・エントリーの追加と削除の中の削除に関するところで述べた説明と同様であるので、ここでは省略する。

この代替キー・エントリーの追加と削除によって、代替キー・ロケーション・テーブルが変更される場合は、追加される代替キー値が当該ブロック中で最大のキー値を持つ場合と、削除される代替キー値が当該ブロック中で最小または最大のキー値を持つ場合である。この場合には、代替キー・ロケーション・テーブルの当該代替キー・ロケーション・テーブル・エントリーの情報の更新を行う。

それに伴って、プライマリー・システム０からアクセラレーター・システム１に対して当該代替キー・ロケーション・テーブル・エントリーの更新情報を送信する。この代替キー・エントリーの追加は、代替キー・テーブルＢ，Ｃに関しても同様に実施する。

代替キー・テーブルＡ，Ｂ，Ｃへの代替キー・エントリーの追加・削除が完了したら、これまでの排他を解除する。

[代替キーによるレコードの削除：アクセラレーター・システムの場合]

アクセラレーター・システム１上で行う場合には、まず、代替キー・ロケーション・テーブルＡＬ１をバイナリー・サーチして、削除するレコードの代替キー値を持つ代替キー・ロケーション・テーブル・エントリーを検索する。
代替キー・ロケーション・テーブル・エントリーを検索した後、プライマリー・システム上の代替キー・ブロック内を検索し、代替キー・エントリーを見つけ出す。代替キー・エントリーが検索できたら、ロケーション・テーブルＬ１を使用して、バイナリー・サーチを行い、レコードが格納されているブロックを検索する。

この動作は、代替キーによるレコードの検索で述べた方法と同様である。

代替キーはノン・ユニークなキーであるため、代替キー値でレコードを検索した場合には、複数のレコードが該当する可能性がある。削除を代替キーで実行する場合に

は、同一の代替キー値を持つ対象レコードを順次読み込んで、その主キー値やデータの内容によって、削除を行うか否かの選択を行うことになる。ここでは、複数のレコードを順次削除するものとする。

検索したロケーション・テーブル・エントリーが、ブロック番号「ｎ」を指していたとする。ブロックはプライマリー・システム０に存在しているので、それに対してアクセスする。ブロック内のレコードに対する検索は、プライマリー・システム０にあるブロック「ｎ」に対して行う。

ブロック「ｎ」内のレコードを調べ、ターゲット・キー値と同じ主キー値を持つレコードを見つける。この後、「主キーによるレコードの削除」の場合と同様に、ブロック内で削除レコードが占めていた領域を詰め、必要に応じてロケーション・テーブル・エントリーの更新を行い、代替キー・エントリーの削除を行う。

このようにして、レコードの削除を行う。一連の動作が完了したら、排他解除を行う。

以上で、アクセラレーター・システム内でのデータ（レコード）の検索、追加、更新、削除に伴うプライマリー・システム内での動作と、それに伴う、アクセラレーター・システムの動作に関して述べた。

[アクセラレーター・システムを複数作成する場合]

以上で、アクセラレーター・システム１が存在する場合に関して述べたが、アクセラレーター・システムを複数作成することが可能である。アクセラレーター・システム２、３、４、・・・・、ｎが存在する場合の動作は、単一のアクセラレーター・システムの場合と殆ど同様である。違いは、プライマリー・システムでロケーション・テーブルまたは代替キー・ロケーション・テーブルに変更が発生した場合に、アクセラレーター・システム１にのみ情報を送信し、情報の更新完了を待つのではなく、すべてのアクセラレーター・システム２、３、・・・・、ｎに情報を送信し、すべてのアクセラレーター・システムでの情報の更新完了を待つ、ということである。

ロケーション・テーブル及び代替キー・ロケーション・テーブルの更新は、プライマリー・システム上で最初に発生するが、そのきっかけとなるのは、プライマリー・

システム上での、データの追加、更新、削除要求の他に、各アクセラレーター・システムからの、データ追加、更新、削除要求である。アクセラレーター・システム１，２、３、・・・・、ｎからのデータ追加、更新、削除要求で、プライマリー・システムのロケーション・テーブル及び代替キー・ロケーション・テーブルの更新が発生すると、その更新情報を、すべてのアクセラレーター・システムに送信し、同期更新を行う。

このように、プライマリー・システムでロケーション・テーブルまたは代替キー・ロケーション・テーブルに変更が在った場合には、プライマリー・システムから総てのアクセラレーター・システムに対して更新情報を送信し、総てのアクセラレーター・システムで必要な更新を即時に行わなければならないため、一見、効率的に劣るように見える。しかしながら、ロケーション・テーブルや代替キー・ロケーション・テーブルが変更されるのは、それらにキー値の最小値と最大値の双方または何れかを持っている場合で、そのキー値が変更される場合であることを考えると、レコードの更新、追加、削除の回数に比較して、相当に少ないものであり、従来の方法のように、完全なミラー・コピーを持つ方法に比較して、データ送信量が大幅に少なく、また、アクセラレーター・システムで必要な格納領域が、プライマリー・システムに比較して大幅に少なくて済むという利点がある。

更に、各アクセラレーター・システムでエントリーに対する検索を独立して行うので、システムの負荷が分散でき、従来の方法に比較して、大幅なスケーラビリティの向上が行える。

[対称型システム]

以上では、プライマリー・システムは、１つのロケーション・テーブルとｋ種類の代替キー・テーブルを持つ構造をしている場合に、この（１＋ｋ）個を組として、アクセラレーター・システムをｎ個作成する構成（対称型アクセラレーター・システム）とすることが可能である。図１４は、プライマリー・システムに対して、対称型アクセラレーター・システムが３つの場合の例である。

[非対称システム]

この方法のほかに非対称アクセラレーター・システムを構成することも可能である。図１５を使用して説明を行う。これは、プライマリー・システムのロケーション・テーブルと、代替キーＡ、Ｂ、Ｃ、・・・に対して、アクセスの頻度により、各アクセラレーター・システムのロケーション・テーブルと代替キー・ロケーション・テーブルの数を決定する方法である。例えば、プライマリー・システムが、ロケーション・テーブルと代替キーＡＬ０、ＢＬ０、ＣＬ０の３種類の代替キー・ロケーション・テーブルを持っている場合について説明する。

アクセラレーター・システム１は、ロケーション・テーブルＬ１と代替キー・ロケーション・テーブルＡＬＡ１、ＡＬＢ１、ＡＬＣ１を保有し、プライマリー・システムと同様の構成である。アクセラレーター・システム２はロケーション・テーブルＬ２と、代替キー・ロケーション・テーブルＡＬＢ２、ＡＬＣ２を保有し、プライマリー・システムと比較すると、代替キー・ロケーション・テーブルＡＬＣ２を保有しない形態となる。この状態を図１５で示している。

同様に、アクセラレーター・システム３はロケーション・テーブル３と代替キー・ロケーション・テーブルＡＬＣ３を保有する。

これは主キーと代替キー各々の使用頻度が分かっていて、その頻度に大きな違いがある場合に有効である。対称アクセラレーター・システムの場合は、あまり使用されないテーブルも保有することになり、格納領域の無駄が発生するからである。

また、アクセラレーターに対するアクセスの統計をとるようにし、主キーや代替キー毎のアクセスが変動する場合には、各アクセラレーターの個数を変更することが好ましい。また、次のように個数変更を自動化することも可能である。

自動化する方式を以下に述べる。

まず、アクセラレーター・システムのキー毎に統計をとる。そして各キー毎のアクセス量を比較する。一定以上のアクセスの場合、非対称アクセラレーターの追加を行う。

例として下記のような、非対称アクセラレーター・システムが設定されているとする。

非対称アクセラレーター・システム１：

ロケーション・テーブルＬ１、
代替キー・ロケーション・テーブルＡＬＡ１、
代替キー・ロケーション・テーブルＡＬＢ１、
代替キー・ロケーション・テーブルＡＬＣ１、

非対称アクセラレーター・システム２：
ロケーション・テーブルＬ２、
代替キー・ロケーション・テーブルＡＬＢ２、
代替キー・ロケーション・テーブルＡＬＣ２、

非対称アクセラレーター・システム３：
ロケーション・テーブルＬ３、
代替キー・ロケーション・テーブルＡＬＣ３、

という構成だった場合、代替キー・テーブルＡのアクセスが増加して、追加が必要だとすると、非対称アクセラレーター・システム２に対して、代替キー・ロケーション・テーブルＡＬＡ２を追加する。非対称アクセラレーター・システム３に追加してもよいが、本事例では、アクセラレーター・システムのうち、番号の若いシステムに追加することとする。追加の方法は、以下に述べるとおりである。

アクセラレーター・システムに代替キー・ロケーション・テーブルＡＬ０を追加するということを、プライマリー・システムまたは状態監視システムなどで決定する。決定後、まず、アクセラレーター・システム２に対して、代替キー・ロケーション・テーブルＡＬＡ２用の領域確保を命ずる。アクセラレーター・システム２では、その情報に基づいて領域の確保を行う。領域の確保が完了したら、プライマリー・システムに対して、完了した旨を送信する。その後、プライマリー・システムは、アクセラレーター・システム２に対して、代替キー・ロケーション・テーブルＡＬＡ０のすべてのエントリーをＡログとして送信する。送信が完了するまでの間に発生する、データ更新に伴うＡログも、同様に送信する。

アクセラレーター・システム２では、Ａログの内容に基づいて、代替キー・テーブ

ル２Ａの該当するエントリーを順次更新する。Ａログの更新がすべて完了した時点で同期が完了したことになり、代替キー・ロケーション・テーブルＡＬＡ２を使用することが可能となる。プライマリー・システムからアクセラレーター・システムに送信を行って、アクセラレーター・システムで代替キー・ロケーション・テーブルＡＬＡ２の作成を行っている間でも、プライマリー・システムやアクセラレーター・システム１での代替キーＡに対するアクセスは可能である。

以上では、プライマリー・システムから、情報を送信する例を示したが、プライマリー・システムの負荷が大きくなりすぎる場合には、アクセラレーター・システム２以外のアクセラレーター・システムから送信するようにしても良い。図１６では、アクセラレーター・システム２に代替キー・ロケーション・テーブルＡＬＡ２の追加が終了した状態を示している。

以上の方式を適用する際に、高速性を確保するためには、各々のアクセラレーターに対してプロセッサーを割り当てることが好ましい。更には、アクセラレーター・システムのロケーション・テーブルと代替キー・テーブルの各々に対して、プロセッサーを割り当てる方式は、より高速性を高める上で好ましい。

アクセラレーター・システムは、プライマリー・システムと同一のハードウエアに内蔵する形式が可能であるが、その他に、アクセラレーター・システム毎に、プライマリー・システムとハードウエアを分離することも可能である。

このようにしてアクセラレーター・システムの数を増やしていくと、テーブルのアクセス数が増加するために、検索等の対象となるブロックのアクセスが増加して、ブロックをアクセスするためのプロセッサーの能力が限界になる可能性があるが、この場合には、ブロックを複数個のバンクに分離し、その各々にプロセッサーを割り当てることによって、負荷分散を図ることが可能である。

[アクセラレーター・システムの変形]

以上で、アクセラレーター・システムの基本的な説明を行った。以上では、プライマリー・システムが１台のハードウエア上に構築し、アクセラレーター・システムは別のハードウエア上に構築する、という概念を用いて説明した。この方が、アクセラレーターの本質的な問題を理解しやすいからである。しかしながら、実際には、プラ

イマリー・システムが１台のハードウエア上に構築されている場合には、プライマリー・システムの負荷が、アクセラレーター・システムに比較して大きくなるため、全体としての処理の力向上が限定される可能性がある。

このような問題を防ぐためには、次のように、プライマリー・システムを分散化させると効果的である。図１７は、その一例である。この場合、プライマリー・システムのロケーション・テーブルとブロックは、プライマリー・システム０－０に、代替キー・テーブルＡ０、Ｂ０、Ｃ０は、プライマリー・システム０－１上に構築されている。

他の方法としては、プライマリー・システムに複数のＣＰＵを割り当てて、ブロック内の検索や更新、アクセラレーター・システムに対する変更情報の送信や、アクセラレーター・システムからの更新完了の受信などを分散化させる、という方法も可能である。

上記以外に、次のような方法も可能である。プライマリー・システムでは、１種類のデータベースが存在するかのようにして説明してきたが、実際の情報処理システムでは、複数種類のデータベースを使用するのが殆どである。このような場合、例えば、データベースαのプライマリー・システムを、ハードウエア０に構築し、データベースβのプライマリー・システムを、ハードウエア１に構築し、データベースγのプライマリー・システムを、ハードウエア２に構築する。そして、データベースαのアクセラレーター・システムを、ハードウエア１と２に構築する。データベースβのアクセラレーター・システムを、ハードウエア０と２に構築する。データベースγのアクセラレーター・システムを、ハードウエア０と１に構築する、というような方法である。

各々のアクセラレーター・システムのＣＰＵが複数持つことが好ましいことを述べたが、、同様にプライマリー・システムも複数のＣＰＵを持つことが好ましい。これは、ロケーション・テーブルや代替キー・ロケーション・テーブルのサーチの他に、ブロックや代替キー・ブロック内サーチやその内容の更新、アクセラレーター・システムとの通信など、様々な負荷が掛かるからである。

［格納領域の優位性］

「データバックアップ・リカバリー方式」で示した、セカンダリー・システムを参照用として用いる場合や、ミラー・ファイルを使用する方式に比較して、格納領域が小さくて済む優位性に関して述べる。

セカンダリー・システムやミラー・ファイルを使用する場合には、現用のシステムと同じ大きさの領域が必要である。一方、アクセラレーター・システムを使用した場合は、ブロック全体の必要領域に対して、ロケーション・テーブルと代替キー・テーブルの大きさは、レコードの長さ、キーの長さ、ブロックの大きさ、ブロックへの格納率、代替キーの種類、などによって条件が異なるため、厳密に述べることは不可能であるが、ファイルに対して、ロケーション・テーブルの大きさは１００分の１程度、代替キー・テーブル１種類の大きさは１０分の１程度であると想定できる。また、代替キー・テーブルに占める代替キー・ロケーション・テーブルの大きさは、１００分の１程度と想定できる。これは、レコード長を４００バイト、ブロック長を４Ｋバイト、ロケーション・テーブル・エントリーの大きさを４０バイト、代替キー・エントリーの大きさを４０バイトと想定した場合である。

このため、セカンダリー・システムやミラー・ファイルを用いる場合に比較して、代替キー・テーブルが５種類ある場合で、６つの対称アクセラレーター・システムを採用した場合とすると、以下のようになり、セカンダリー・システムやミラー・ファイルを使用した場合に比較し、約５分の１の格納容量で済むことになる。アクセラレーター・システムの数を増やした場合には、更に、効果が増すことが容易に理解できる。

プライマリー・システムの大きさ（ブッロクの大きさを１とした場合の全体の大きさ）は、次のように、

１＋１／１００＋５×（１／１０）＝１５１／１００

である。

６つのセカンダリー・システムまたはミラー・ファイルを使用した場合の領域は、次のようになり、

（１５１／１００）×６＝９０６／１００

であるから、プライマリー・システムの６倍になる。

６つの対称アクセラレーターを用意した場合、
（１／１００＋（１／１０）＊（１／１００））×６＝６６／１０００
となる。

よって、両者の必要領域の比は、
（９０５／１００）：（６５／１０００）＝１３９：１
となる。

つまり、アクセラレーター・システムを使用した場合には、セカンダリー・システムやミラー・ファイルを使用する場合に比較して、１％程度の領域を使用するだけで済むことを示している。

[バックアップ・リカバリー方式との連携]
次に、「データバックアップ・リカバリー方式」との同時使用が可能であることを説明する。
バックアップ・リカバリーは同期密結合方式と、非同期疎結合方式がある。同期密結合方式のバックアップ・リカバリーと、アクセラレーター・システムにおける、ロケーション・テーブルおよび／または代替キー・ロケーション・テーブルの更新は、同期を取って実行する。しかしながら、バックアップ・リカバリーは、ロケーション・テーブルおよび／または代替キー・ロケーション・テーブルの更新よりも、ブロックおよび／または代替キー・ブロック（何れもオーバーフロー・ブロックを含む）の更新機会の方がはるかに多いので、バックアップ・リカバリーにおけるセカンダリー・システムの更新の都度、アクセラレーター・システムの更新が必要になるものでは無い。
バックアップ・リカバリーが非同期疎結合方式の場合には、アクセラレーター・システムのみ、同期密結合方式で実行する。

[再編成システムとの連携]

次に、「データベース再編成システム」との同時使用が可能であることを説明する。

プライマリー・システムで再編成を行う場合には、同時にアクセラレーター・システムにも、現用のロケーション・テーブルに対して、新規のロケーション・テーブルを用意して、ロケーション・テーブルとブロックの再編成を実行する。プライマリー・システムとアクセラレーター・システムの再編成は同期をとって実行する。

アクセラレーター・システムでも、再編成ポインターを保持し、プライマリー・システムの再編成ポインターと同期させる。同期させるというのは、再編成ポインターが指しているロケーション・テーブル・エントリーの位置が、プライマリー・システムとアクセラレーター・システムで同じにするということである。

検索等のターゲット・キー値が再編成ポインターが指しているロケーション・テーブルの主キー値の最小値よりより小さい場合には、新規のロケーション・テーブルを使用し、検索等のターゲット・キー値が再編成ポインターが指しているロケーション・テーブルの主キー値の最小値以上である場合には、現用のロケーション・テーブルを使用して、バイナリー・サーチを実行し、目的のレコードを検索する。

同様に、プライマリー・システムで代替キー・テーブルの再編成を行う場合には、同時にアクセラレーター・システムにも、現用の代替キー・ロケーション・テーブルに対して、新規の代替キー・ロケーション・テーブルを用意して、代替キー・ロケーション・テーブルと代替キー・ブロックの再編成を実行する。プライマリー・システムとアクセラレーター・システムの再編成は同期をとって実行する。

アクセラレーター・システムでも、再編成ポインターを保持し、プライマリー・システムの再編成ポインターと同期させる。同期させるというのは、再編成ポインターが指しているロケーション・テーブル・エントリーの位置が、プライマリー・システムとアクセラレーター・システムで同じにするということである。

検索等のターゲット・キー値が再編成ポインターが指している代替キー・ロケーション・テーブルの代替キー値の最小値よりより小さい場合には、新規の代替キー・ロケーション・テーブルを使用し、検索等のターゲット・キー値が再編成ポインターが指している代替キー・ロケーション・テーブルの主キー値の最小値以上である場合には、現用の代替キー・ロケーション・テーブルを使用して、バイナリー・サーチを実行し、目的の代替キー・エントリーを検索する。

代替キー・エントリーが見つかったら、そのエントリーが持っている情報により、ロケーション・テーブルからブロックを探し、レコードの検索を行う。

このように行うことで、再編成システムとの連携が可能である。

「データ格納検索方式」では、ロケーション・テーブルの構造として、ロケーション・テーブルの番号（＝プライマリー・ブロックの番号）と、ブロックの位置を保持し、ブロック中のレコードの主キー値の最小と最大のものを、両方またはどちらか一方を保持することも可能としてある。主キー値をロケーション・テーブルで保持しない場合は、ロケーション・テーブルに対する更新が減少するが、バイナリー・サーチでブロックを探す都度、ブロックを読まないと目的のブロックであるか否かの判定ができないため、ブロックに対する負荷が増加するので、本発明に適用する場合には、ロケーション・テーブルにキー値を保持する方式が好ましい。キー値は最小と最大の両方を保持している場合には、そのエントリーを読むだけで目的のブロックであるか否かの判定が可能である。どちらか一方を保持している場合には、そのエントリーの後、または、前のエントリーを読むことにより、解決が行える。

ロケーション・テーブルは、再編成を除いて、ロケーション・テーブルのエントリーが保持しているブロックの位置が変更されることはなく、ブロックに含まれるレコードの最小値、最大値の双方または何れか一方が変更された場合に変更が発生するため、旧来の方法によるインデックスに比較して、変更の発生ははるかに少ない。旧来の方法では、ブロックに相当するものの分割が発生して、ロケーション・テーブルのエントリーに相当するものの数が増減するため変更が多いのである。また、旧来の方法でのインデックスでは、階層構造をとっているため、上位の階層の変更が発生するケースが頻繁にあり、これも、旧来の方法でインデックスの変更が頻繁に発生する原因となっていた。

本発明を適用すると、アクセラレーター・システムが複数存在することになり、数が増加すると、各々に対してプライマリー・システムから変更を通知し、その結果を待つための負荷が増大する。これを軽減する方法として、一斉同報方式を用いて、セカンダリー・システムに対して変更を通知する方式が有利である。

セカンダリー・システムでセカンダリー・ロケーション・テーブルの更新が完了した通知は各々のセカンダリー・システムから個別にプライマリー・システムに対して

送信する。

各アクセラレーター・システムに対して、どのように要求を割振るかに関して述べる。このデータ格納・検索システムに対する要求は、ＳＱＬのような形式の他、Read/Writeインターフェイスも可能であるが、それらは、最終的には、次のような情報となる。

１．対象となるキーの種類（主キー、代替キーＡ、Ｂ、Ｃ・・・）。
２．アクセスの種類（Ｒｅａｄ、Ｗｒｉｔｅ、Ｒｅｒｉｔｅ、Ｄｅｌｅｔｅ、Ｉｎｓｅｒｅｔ　など）。
３．要求キー値。
４．データ内容（Ｉｎｓｅｒｔ　の場合）。

これらの情報が一組となって、データ格納・検索システムに対しての処理要求となる。振り分けには、図２０、図２１、図２２を用いて説明する。図２０および図２１では、インターネットを使用して、端末から処理要求が送信される図となっているが、インターネットを使用せず、専用線やＬＡＮを用いる構成でも適用可能である。

図２０に示した図は、処理要求を振り分けるのに、ロード・バランサーを使用する方式を示したものである。ロード・バランサーとは、複数のサーバーが、見かけ上、同じＩＰアドレスを保有しているかのように見せかけ、多数の端末からの接続を、複数のサーバーに振り分けるためのものである。この場合、処理サーバー（一般的には、アプリケーション・サーバーと呼ばれる場合が多い）と、プライマリー・システム、または、アクセラレーターは固定的に接続される。

対称型アクセラレーター・システムを使用する場合には、ロード・バランサーで問題ないが、非対称型アクセラレーター・システムを使用する場合には、ロード・バランサーでは稼動させることができない。何故ならは、特定の代替キーを持っていないアクセラレーター・システムが存在するためである。このような場合には、図２１に示すような処理要求振り分けシステムを使用すると都合がよい。また、図２２は処理要求振り分けシステムの動作を詳細に説明するためのものである。処理要求振り分けシステムは、ロード・バランサーを使用する場合のように、固定的な接続とせず、処

理要求毎に、プライマリー・システムとアクセラレーターを選択する方式である。

図２２中の処理要求には、一連の番号と、キーの種類のみを抜き出して示してある。その他の情報は、これに付属しているものとする。処理要求は、通常用いられているようにキュー構造をするのが好ましい。

図２２には処理システム・テーブルが４つ記載されている。各々は、ロケーション・テーブル用の処理システム・テーブルＬ、代替キー・テーブルＡ用の処理システム・テーブルＡ、代替キー・テーブルＢ用の処理システム・テーブルＢ、代替キー・テーブルＣ用の処理システム・テーブルＣ、となっている。

各処理システム・テーブルのエントリー（処理システム・テーブル・エントリー）は、各アクセラレーター・システムに対応しており、図２２の場合は、ロケーション・テーブルのアクセラレーター・システムが１９個、代替キー・テーブルＡのアクセラレーター・システムが１４個、代替キー・テーブルＢのアクセラレーター・システムが１１個、代替キー・テーブルＣのアクセラレーター・システムが１７個、存在する場合を示している。

各処理システム・テーブルのエントリーは、処理システム番号と処理中フラッグから構成されている。処理システム番号は、ゼロはプライマリー・システムを、１からの正の数字は、アクセラレーター・システムを示している。処理中フラッグは、そのシステムが処理要求を処理中であるか、非処理状態であるかを識別するためのものである。

図２２中に白抜き矢印があるが、これは、各処理システム・テーブルの処理を、プライマリー・システムまたはアクセラレーター・システムのうちのどれが行っているか、システムの最新番号を示している。ゼロはプライマリー・システムを、１からの正の数字は、アクセラレーター・システムを示している。この矢印を、処理システム・ポインターと呼ぶ。この処理システム・ポインターは、テーブル（キー）の種類毎に保持する必要があるため、ロケーション・テーブル用の処理システム・ポインターＬ、代替キー・テーブルＡ用の処理システム・ポインターＡ、代替キー・テーブルＢ用の処理システム・ポインターＢ、代替キー・テーブルＣ用の処理システム・ポインターＣ、などとなっている。

処理システム・ポインターは、初期は、各処理システム・テーブルの先頭を指して

いるが、処理要求が割り当てられる度に、次のエントリーを指すようにする。各処理システム・テーブルの最終エントリーの次を指す場合には、先頭に戻ることとする。

このようにして、各アクセラレーターに、処理要求が均等に割り当てられるようにしている。

図２２は処理要求割り当て後を示しているため、各、処理システム・ポインターの位置が、移動した後の状態を示している。この図では、システム番号が順番に付番されているが、間をとばして付番することも可能である。

処理要求の、１３８はロケーション・テーブルに対する処理要求である。この処理要求が来た時点では、処理中の処理要求が無かったものとする。処理要求１３８は、図中の処理システム・テーブルＬ（ロケーション・テーブルと書かれた表）を参照する。処理中の処理要求が無い時点で発生した処理要求であるので、処理要求振り分けシステムでは、処理システム・テーブルＬの処理システム番号：０のエントリーに対して、処理中のフラッグを立てる。次に、処理システム・ポインターを処理システム番号：１に移動する。次に、処理要求１３８にシステム番号：０を割り振り、処理要求１３８をプライマリー・システムに送信する。

プライマリー・システムでは、処理要求１３８に基づき、ロケーション・テーブルに対する処理を行い、処理が完了したら、振り分けシステムに対して、処理要求１３８が完了したことを通知する。振り分けシステムでは、プライマリー・システムからの完了通知であるので、処理システム・テーブルＬの処理システム番号：０の処理中のフラッグを、非処理にする。

処理要求：１３９は代替キー・テーブルＡに対する要求であるので、処理システム・テーブルＡを参照する。処理システム・ポインターＡは、処理システム・テーブルの先頭を指しているため、処理要求１３９は、プライマリー・システムに割り当てる。

この割り当て作業は、処理要求：１３８の完了を待つ必要はなく、処理要求：１３８の割り当てを完了したら、即座に実行することが可能であるし、そうすることにより、高速な処理が行える。

同様に、処理要求：１４０は代替キー・テーブルＢに対する処理要求であるので、処理システム・テーブルＢを参照する。処理システム・ポインターＢは、処理システム・テーブルの先頭を指しているため、処理要求１４０は、プライマリー・システム

に割り当てる。

その後の、動作は、処理要求：１３８と同様である。処理システム・ポインターは、処理システム・テーブルの最後までポイントしたら、先頭に戻るように設計しておく。先頭に戻った場合に、処理システム・ポインターが指している処理システム番号が処理中の場合は、次の要求を待ちにするか、または、次々とシステムを検索し、処理中で無いシステムを見つけて、要求をそのシステムに割り当てるとともに、処理システム・ポインターをそこまで移動するという方法も可能である。後者の方法を採用する場合には、一時的にループする可能性が存在するが、処理システム・テーブルに割り当てられた処理要求の何れかが完了すれば、処理システム・テーブルの当該システム＃の処理中のフラッグが、非処理状態になるため、無限ループは発生しない。また、処理システム・ポインターが処理システム・テーブルを１周した場合には、一定時間待ってから、再度、非処理状態の処理システム・テーブル・エントリーを探す、という方法をとることも可能である。また、このような状態が発生した場合には、特定のキーのアクセラレーター・システムを追加する、ということも有効である。

上記の他に、アクセラレーター・システムをソフトウエアとして構築する場合の例を、図２３を用いて説明する。プライマリー・システムとアクセラレーター・システムは、同一のハードウエアに格納されており、このハードウエアは、複数のＣＰＵを備えている。各々のＣＰＵは、ＯＳによってプライマリー・システムやアクセラレーター・システムに割り当てられるようになっている。プライマリー・システムでは、ロケーション・テーブルと代替キー・テーブルを保有しており、この他に、変更情報送信機構を保有することが好ましい。アクセラレーター・システムは、フランド・ロケーション・テーブルとフランド代替キー・ロケーション・テーブルを保有するが、非対称システムの場合は、プライマリー・システムとアクセラレーター・システムの構成が異なっても良い。

請 求 の 範 囲

1． 主キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、各プライマリー・ブロックのアドレスが記載されたロケーション・テーブル・エントリーを連続領域に有するロケーション・テーブルとを保有するプライマリー・システムを備え、

各プライマリー・ブロックのアドレスが記載されたフランド・ロケーション・テーブル・エントリーを連続領域に有するフランド・ロケーション・テーブルを保有するアクセラレーター・システムを備えたことを特徴とする、データベースのアクセラレーター。

2． 請求項１に記載のデータベースのアクセラレーターにおいて、

プライマリー・システムは、少なくとも１つのＣＰＵを備え、

アクセラレーター・システムは、少なくとも１つのＣＰＵを備えたことを特徴とするアクセラレーター・システム。

3． 請求項１に記載のデータベースのアクセラレーターにおいて、

プライマリー・システムとアクセラレーター・システムは、複数のＣＰＵを共有することを特徴とするアクセラレーター・システム。

4． 主キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、各プライマリー・ブロックのアドレスが記載されたロケーション・テーブル・エントリーを連続領域に有するロケーション・テーブルと、変更情報送信機構とを保有するプライマリー・システムを備え、

各プライマリー・ブロックのアドレスが記載されたフランド・ロケーション・テーブル・エントリーを連続領域に有するフランド・ロケーション・テーブルと、変更情報反映機構とを保有するアクセラレーター・システムを備えたことを特徴とする、データベースのアクセラレーター。

5． 請求項４に記載のデータベースのアクセラレーターにおいて、
プライマリー・システムは、少なくとも１つのＣＰＵを備え、
アクセラレーター・システムは、少なくとも１つのＣＰＵを備えたことを特徴とするアクセラレーター・システム。

6． 主キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、各プライマリー・ブロックのアドレスが記載されたロケーション・テーブル・エントリーを連続領域に有するロケーション・テーブルとを保有するプライマリー・システムを備え、
各プライマリー・ブロックのアドレスが記載されたフランド・ロケーション・テーブル・エントリーを連続領域に有するフランド・ロケーション・テーブルを保有するアクセラレーター・システムを備えた、
アクセラレーター・システムにおける主キーによるアクセスは、フランド・ロケーション・テーブルをバイナリー・サーチし、フランド・ロケーション・テーブル・エントリーが示す結果により、プライマリー・システムのブロックをアクセスすることを特徴とするデータベースのアクセラレーター。

7． 請求項６に記載のデータベースのアクセラレーターにおいて、
プライマリー・システムは、少なくとも１つのＣＰＵを備え、
アクセラレーター・システムは、少なくとも１つのＣＰＵを備えたことを特徴とするアクセラレーター・システム。

8． 主キーと代替キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、代替キーと主キーからなる代替キー・エントリーと、代替キー・エントリーを含む代替キー・ブロックと、代替キー・ロケーション・テーブル・エントリーを連続領域に有する代替キー・ロケーション・テーブルを保有するプライマリー・システムを備え、
フランド代替キー・ロケーション・テーブル・エントリーを連続領域に有するフランド代替キー・ロケーション・テーブルを保有するアクセラレーター・システムを備え

たことを特徴とするデータベースのアクセラレーター

９．　請求項８に記載のデータベースのアクセラレーターにおいて、
プライマリー・システムは、少なくとも１つのＣＰＵを備え、
アクセラレーター・システムは、少なくとも１つのＣＰＵを備えたことを特徴とするアクセラレーター・システム。

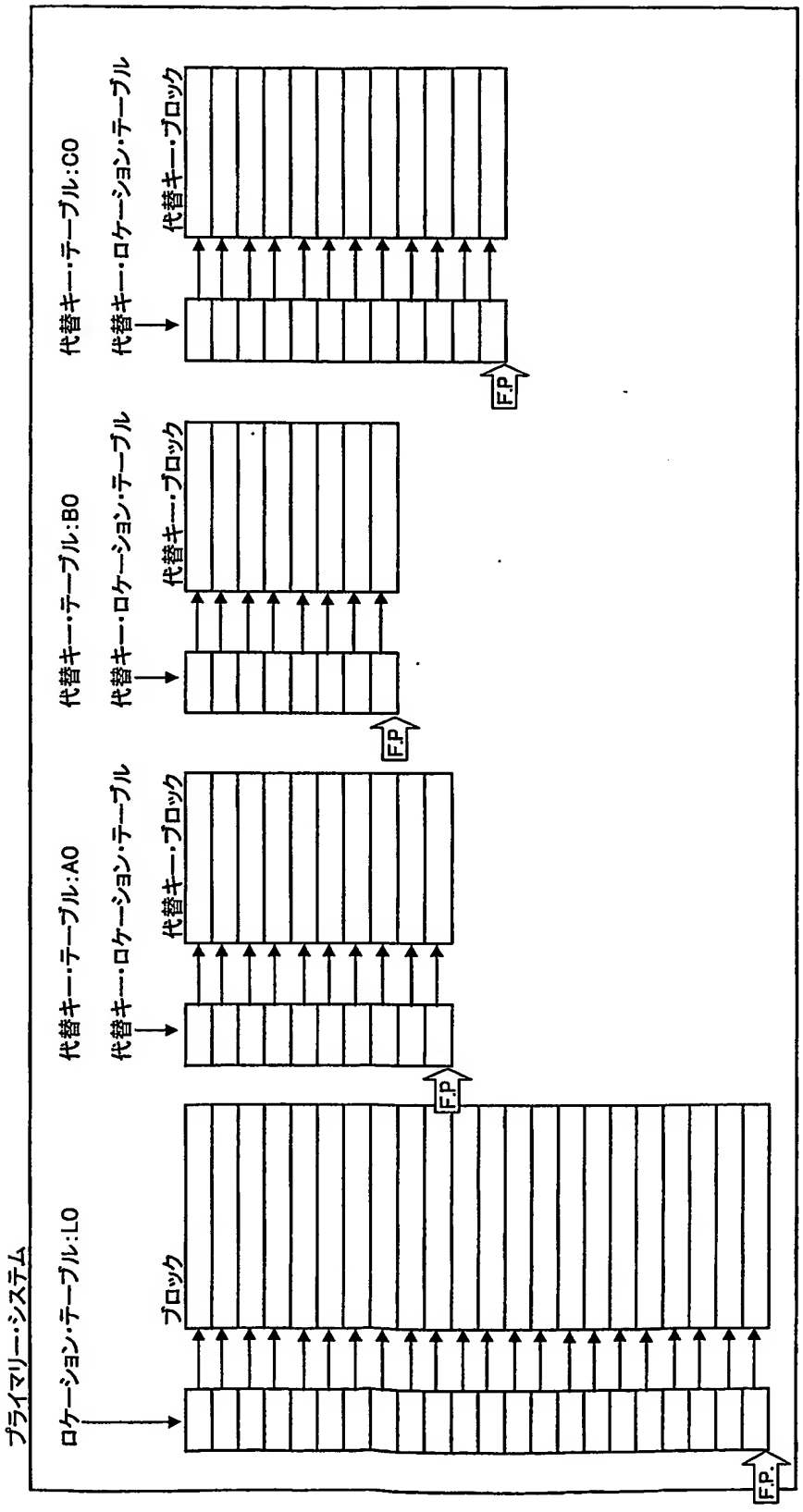
１０．　請求項８に記載のデータベースのアクセラレーターにおいて、
プライマリー・システムとアクセラレーター・システムは、複数のＣＰＵを共有することを特徴とするアクセラレーター・システム。

１１．　主キーと代替キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、代替キーと主キーからなる代替キー・エントリーと、代替キー・エントリーを含む代替キー・ブロックと、代替キー・ロケーション・テーブル・エントリーを連続領域に有する代替キー・ロケーション・テーブルとを保有するプライマリー・システムを備え、
各代替キー・ブロックのアドレスが記載されたフランド代替キー・ロケーション・テーブル・エントリーを連続領域に有するフランド代替キー・ロケーション・テーブルと、変更情報反映機構とを保有するアクセラレーター・システムを備えたことを特徴とする、データベースのアクセラレーター

１２．　主キーと代替キーを含むデータ項目を持つデータレコードと、データレコードを主キーの順に格納するプライマリー・ブロックと、代替キーと主キーからなる代替キー・エントリーと、代替キー・エントリーを含む代替キー・ブロックと、代替キー・ロケーション・テーブル・エントリーを連続領域に有する代替キー・ロケーション・テーブルとを保有するプライマリー・システムを備え、
各代替キー・ブロックのアドレスが記載されたフランド・ロケーション・テーブル・エントリーを連続領域に有するフランド・ロケーション・テーブルを保有するアクセラレーター・システムを備え、

アクセラレーター・システムにおける代替キーによるアクセスは、フランド代替キー・ロケーション・テーブルをバイナリー・サーチし、フランド代替キー・ロケーション・テーブル・エントリーが示す結果により、プライマリー・システムの代替キー・ブロックをアクセスすることを特徴とするデータベースのアクセラレーター。

図1


プライマリー・システム
ロケーション・テーブル:LO
ブロック
F.P.
代替キー・テーブル:AO
代替キー・ロケーション・テーブル
代替キー・ブロック
F.P
代替キー・テーブル:BO
代替キー・ロケーション・テーブル
代替キー・ブロック
F.P
代替キー・テーブル:CO
代替キー・ロケーション・テーブル
代替キー・ブロック
F.P

【図2】

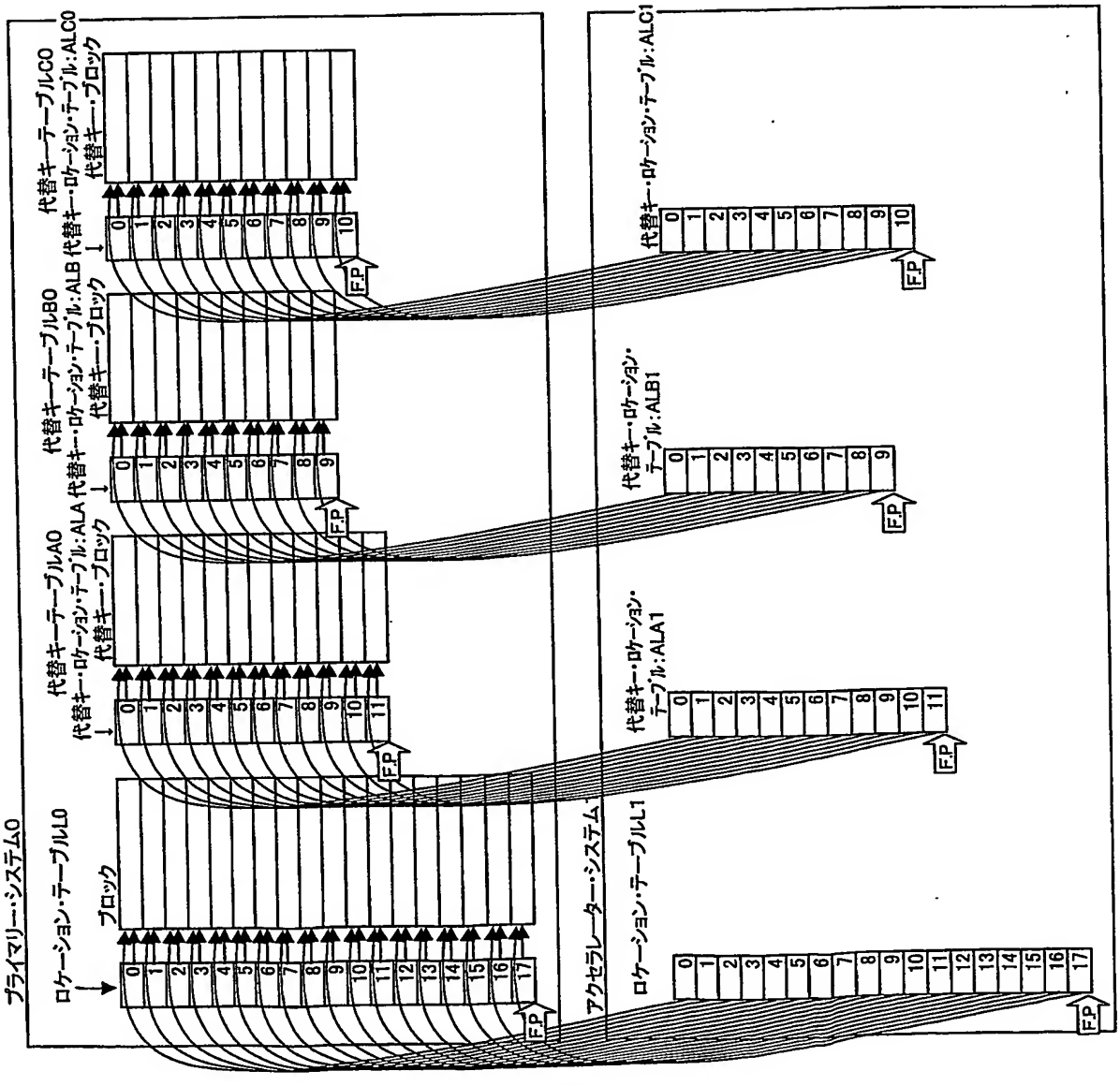

プライマリー・システム0
ロケーション・テーブル:L0
ブロック
代替キー・ロケーション・テーブル:ALA0
代替キー・ブロック
代替キー・テーブル:ALA0
代替キー・ロケーション・テーブル:ALB0
代替キー・ブロック
代替キー・テーブル:ALB0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
代替キー・テーブル:ALC0
F.P
アクセラレーター・システム1
ロケーション・テーブル:L1
代替キー・ロケーション・テーブル:ALA1
代替キー・ロケーション・テーブル:ALB1
代替キー・ロケーション・テーブル:ALC1

図3

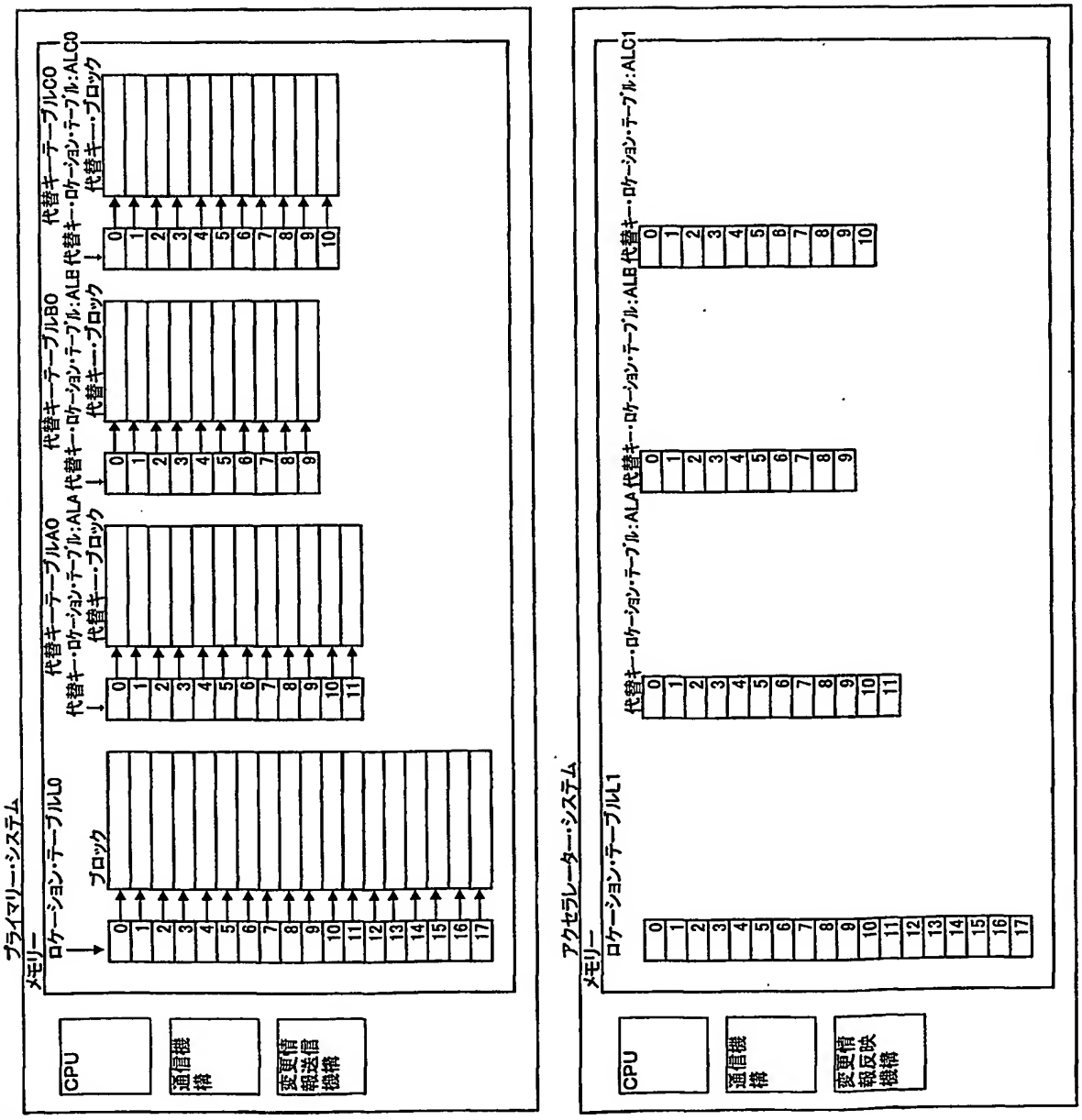
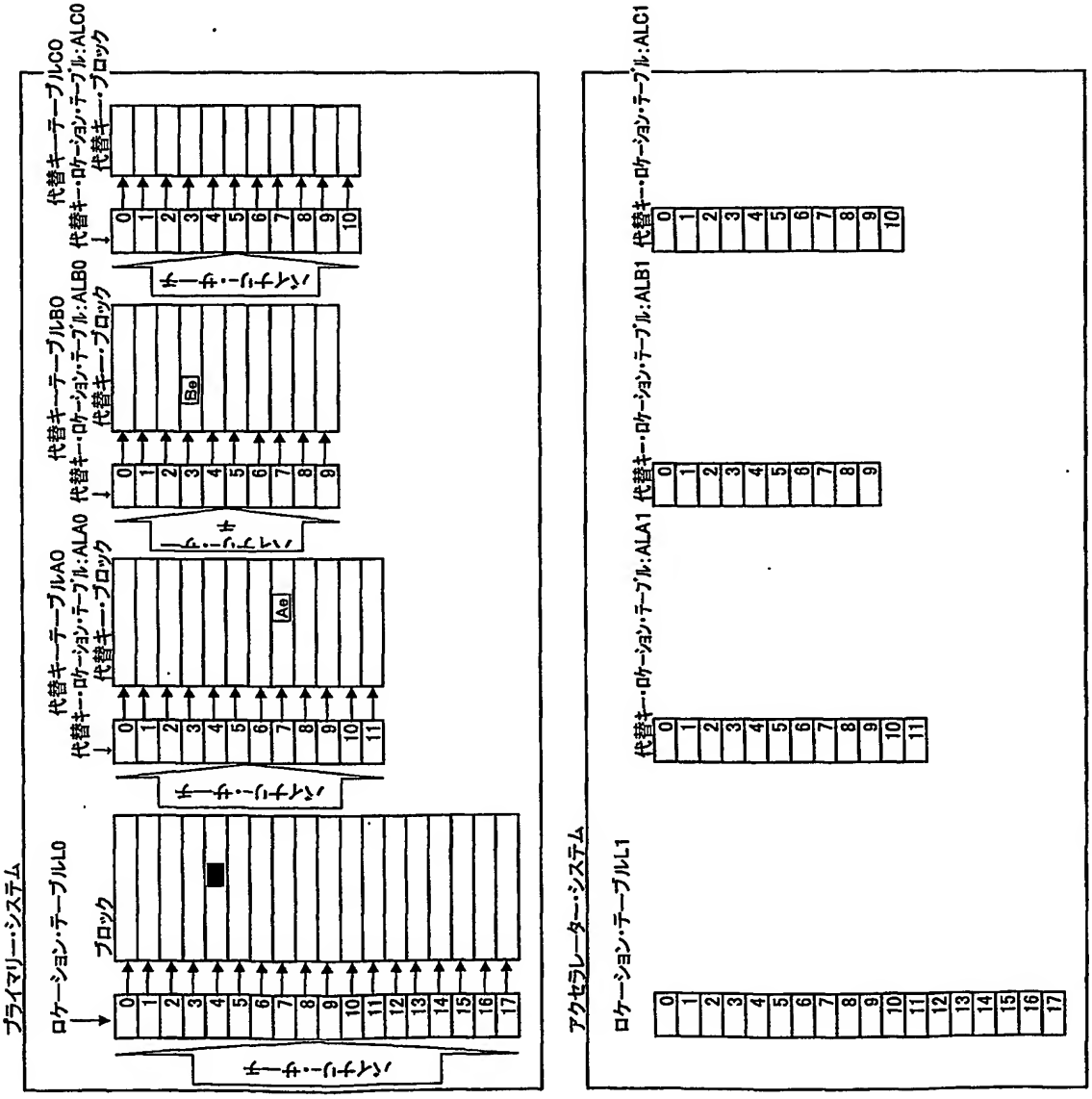

プライマリー・システム
メモリー
ロケーション・テーブルL0
ブロック
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キー・テーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キー・テーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
CPU
通信機構
変更情報送信機構
アクセラレーター・システム
メモリー
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
CPU
通信機構
変更情報反映機構

図4


プライマリー・システム
ロケーション・テーブルL0
ブロック
インデックス・サーチ
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA0
代替キー・ブロック
Ae
代替キーテーブルB0
代替キー・ロケーション・テーブル:ALB0
代替キー・ブロック
Be
代替キーテーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA1
代替キー・ロケーション・テーブル:ALB1
代替キー・ロケーション・テーブル:ALC1

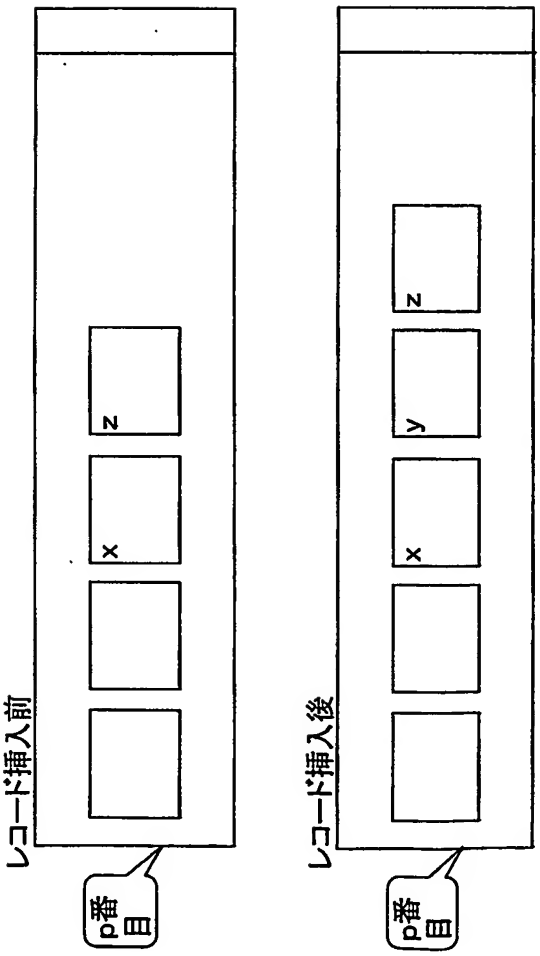


図5

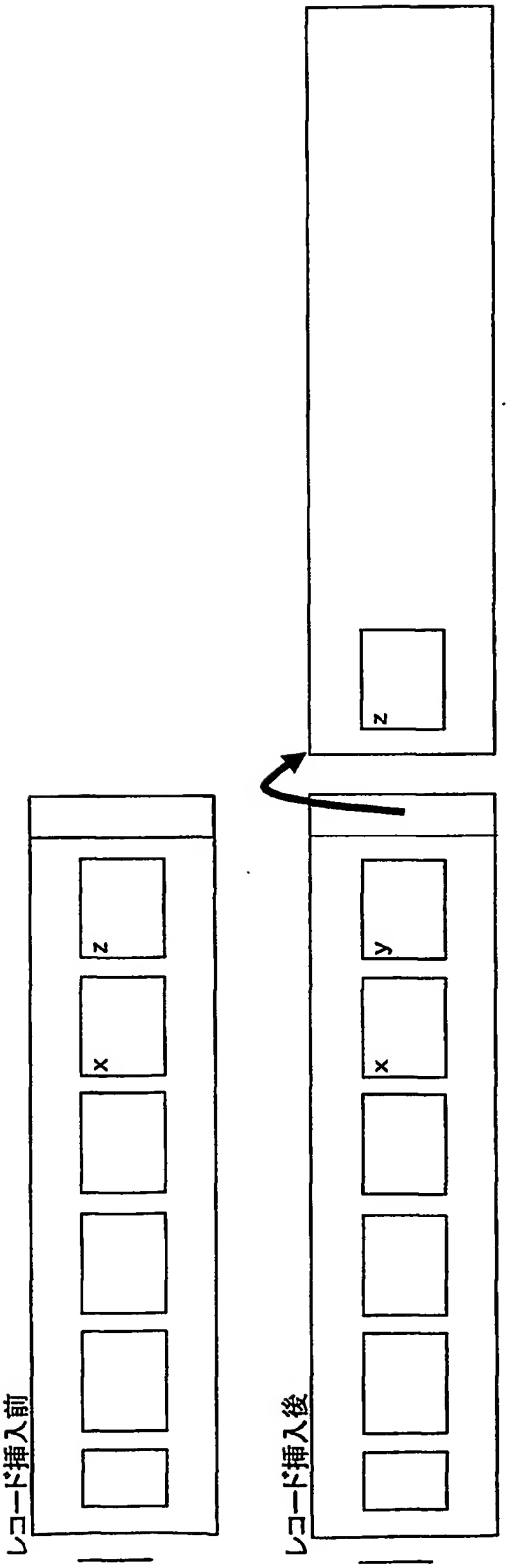
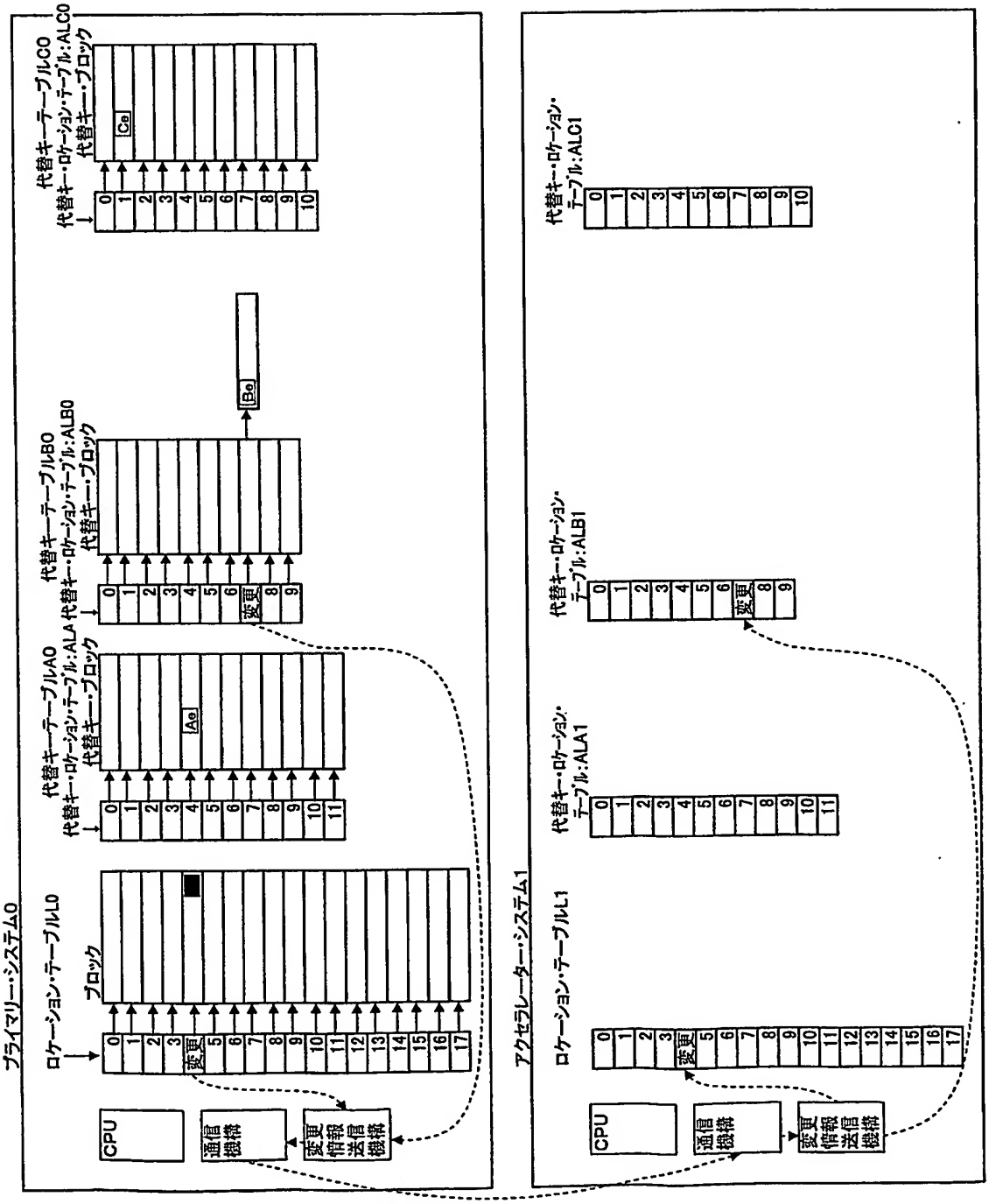


図6

図7


プライマリー・システム0
ロケーション・テーブルL0
ブロック
変更
CPU
通信機構
変更情報送信機構
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
Ae
代替キー・テーブルB0
代替キー・ロケーション・テーブル:ALB0
代替キー・ブロック
Be
代替キー・テーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
Ce
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA1
代替キー・ロケーション・テーブル:ALB1
代替キー・ロケーション・テーブル:ALC1

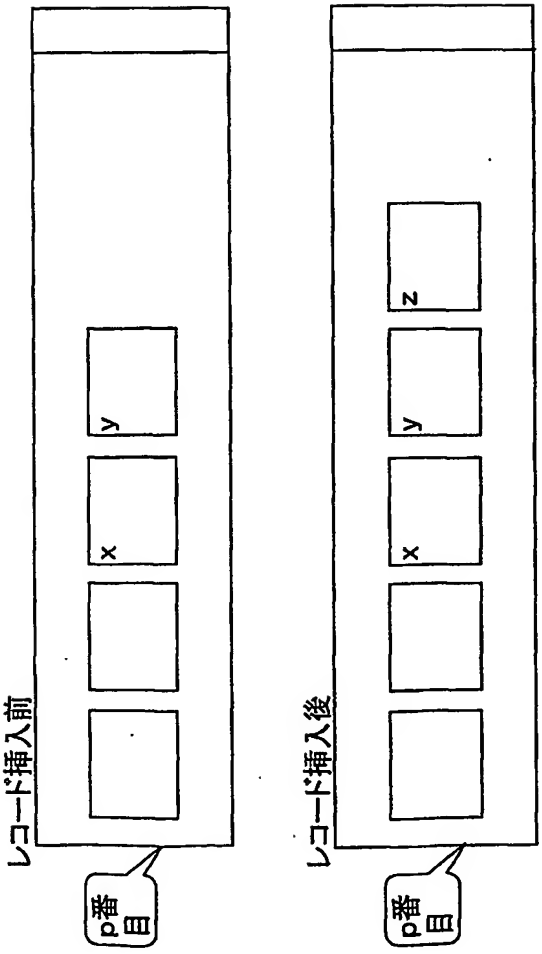


図8

図9

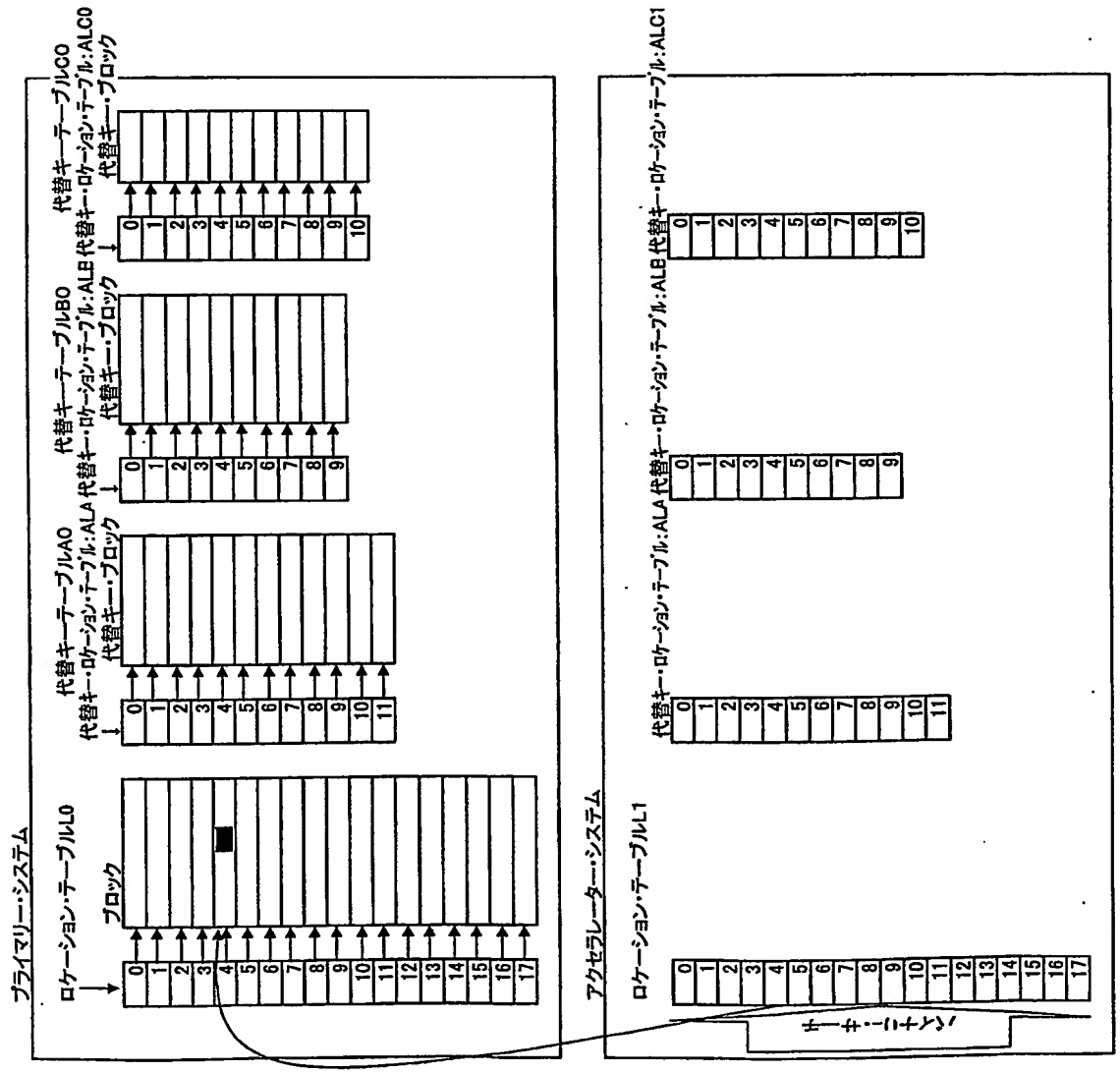

プライマリー・システム
ロケーション・テーブルL0
ブロック
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キーテーブルA0
代替キー・ロケーション・テーブル:ALB
代替キーテーブルB0
代替キーテーブルC0
アクセラレーター・システム
ロケーション・テーブルL1
代替キーテーブルC1
インデックス

図10

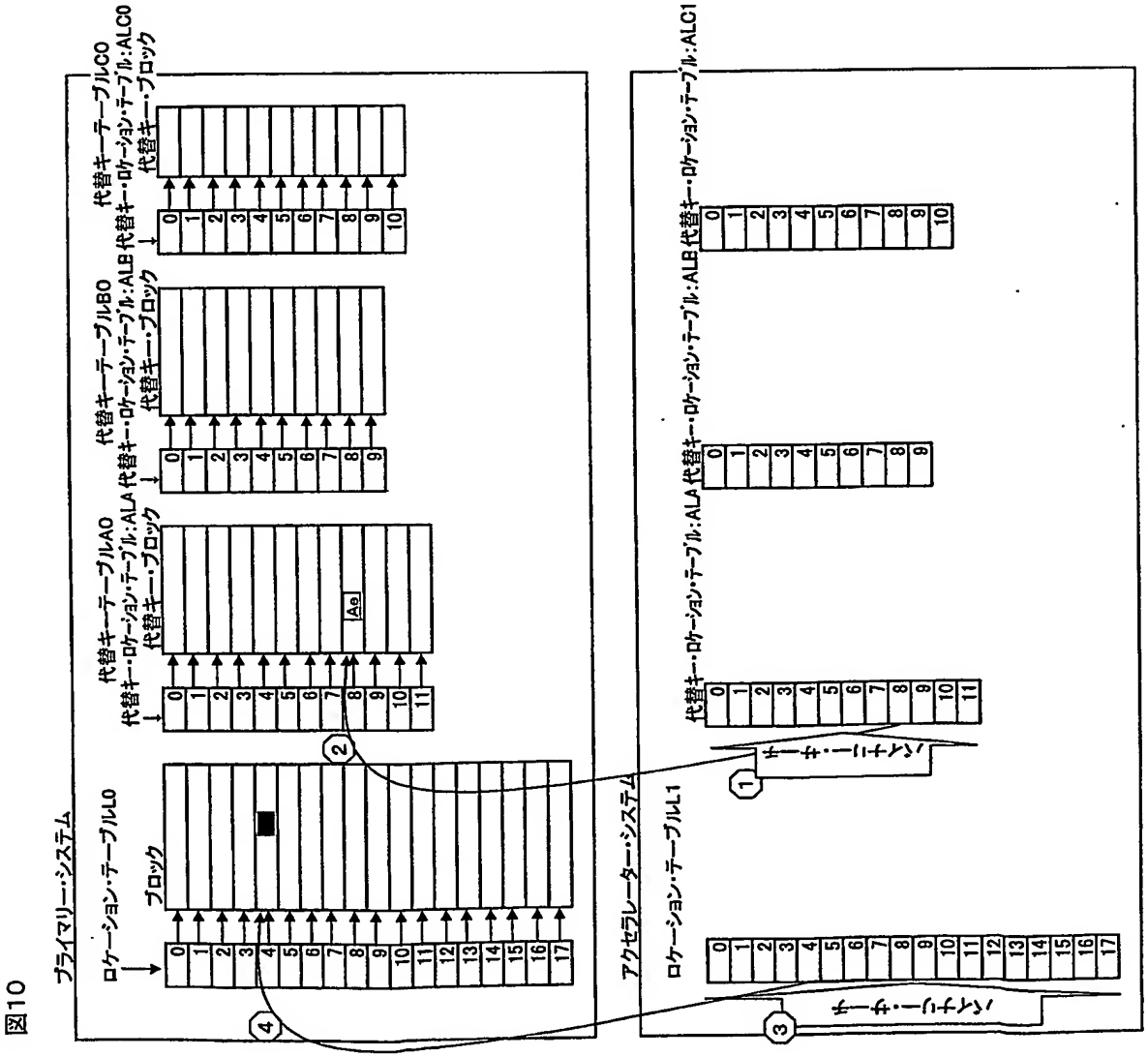

プライマリー・システム
ロケーション・テーブルL0
ブロック
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
Ae
代替キー・テーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キー・テーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
インデックス
インデックス
1
2
3
4

図11

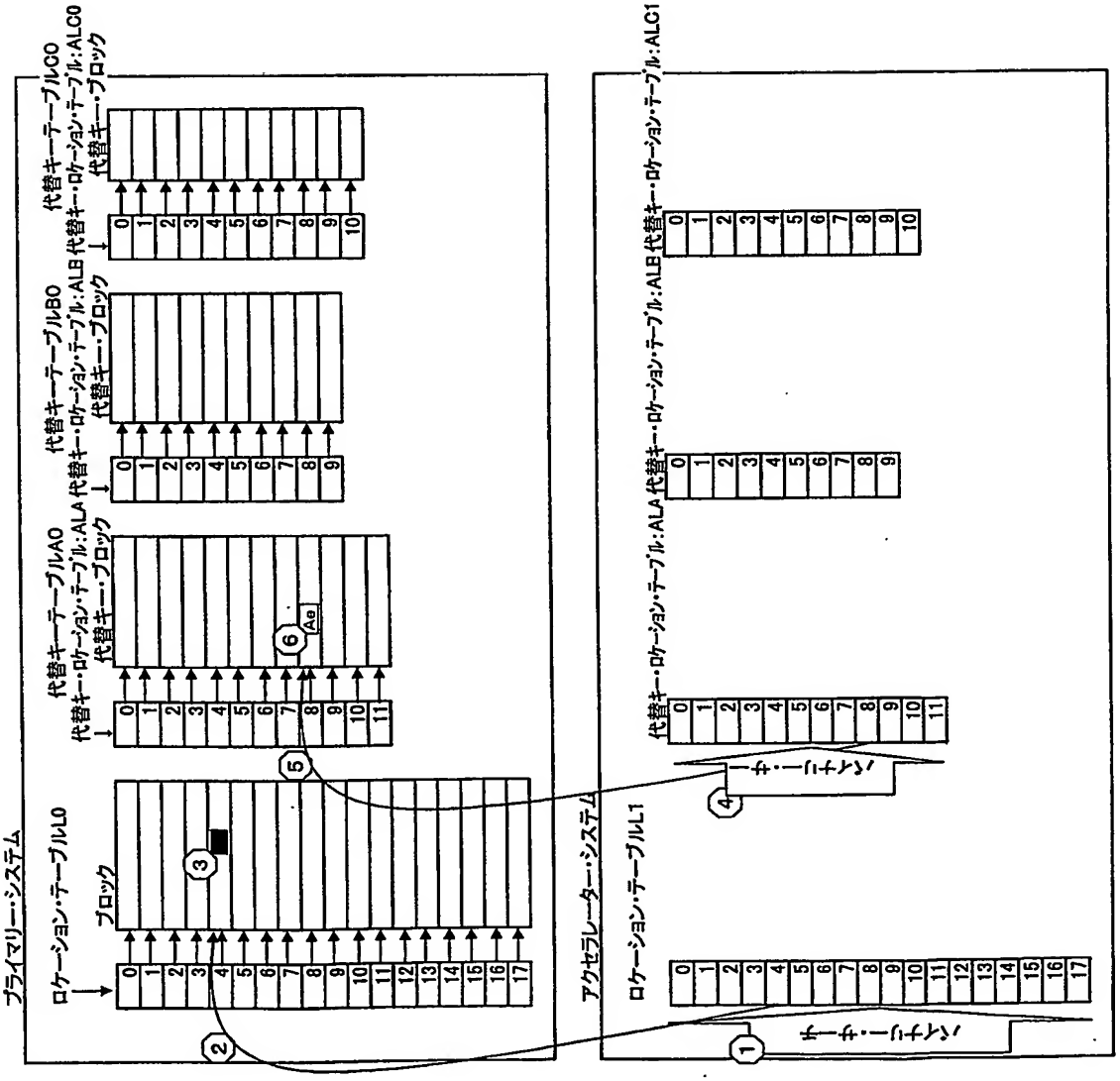

プライマリー・システム
ロケーション・テーブルL0
ブロック
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キー・テーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キー・テーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
インデックス
インデックス
Ae
1
2
3
4
5
6

図12

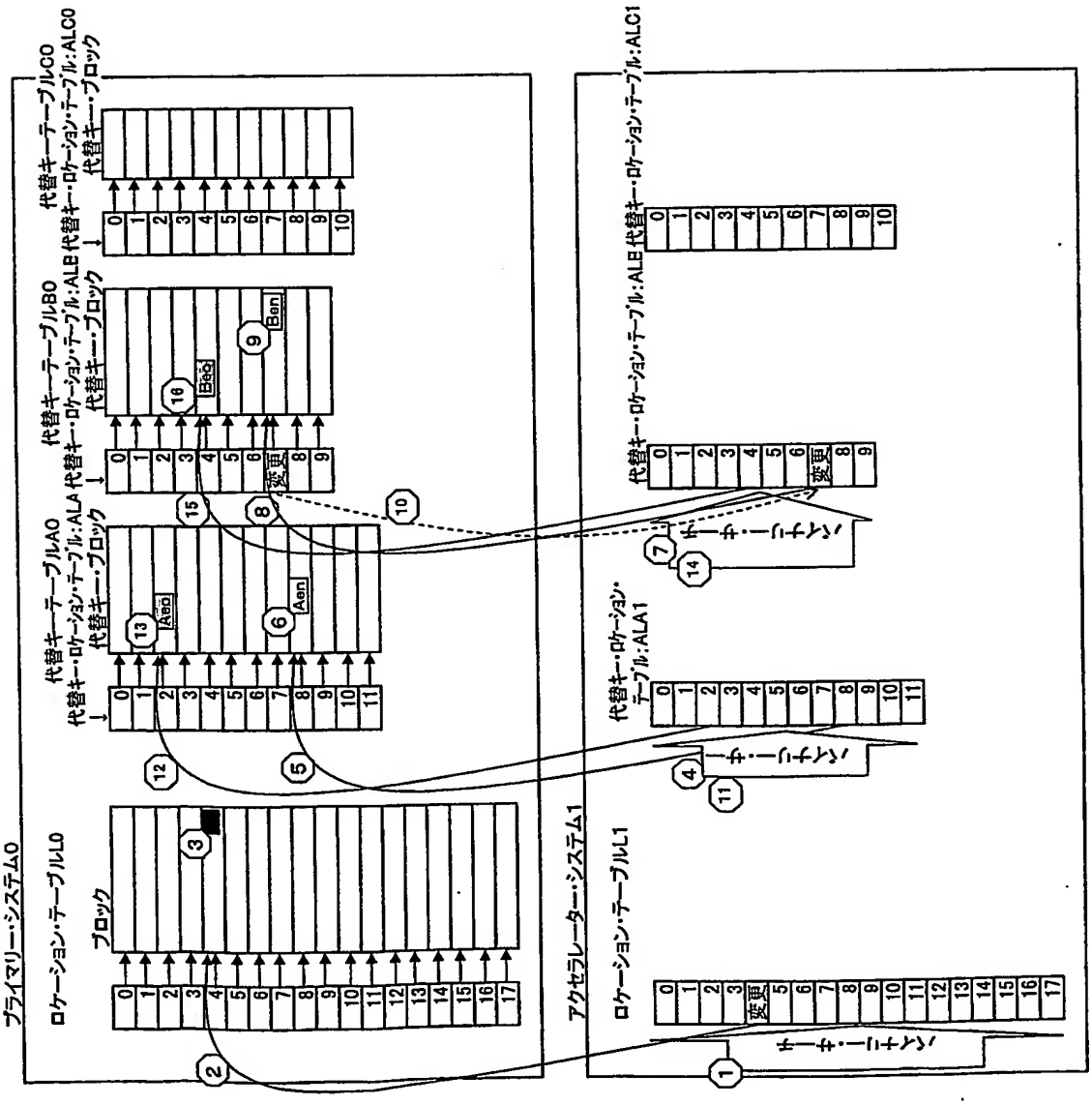

プライマリー・システム0
ロケーション・テーブルL0
ブロック
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キー・テーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キー・テーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA1
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
変更
Aen
Aea
Ben
Bea
インナリー・キー

図13

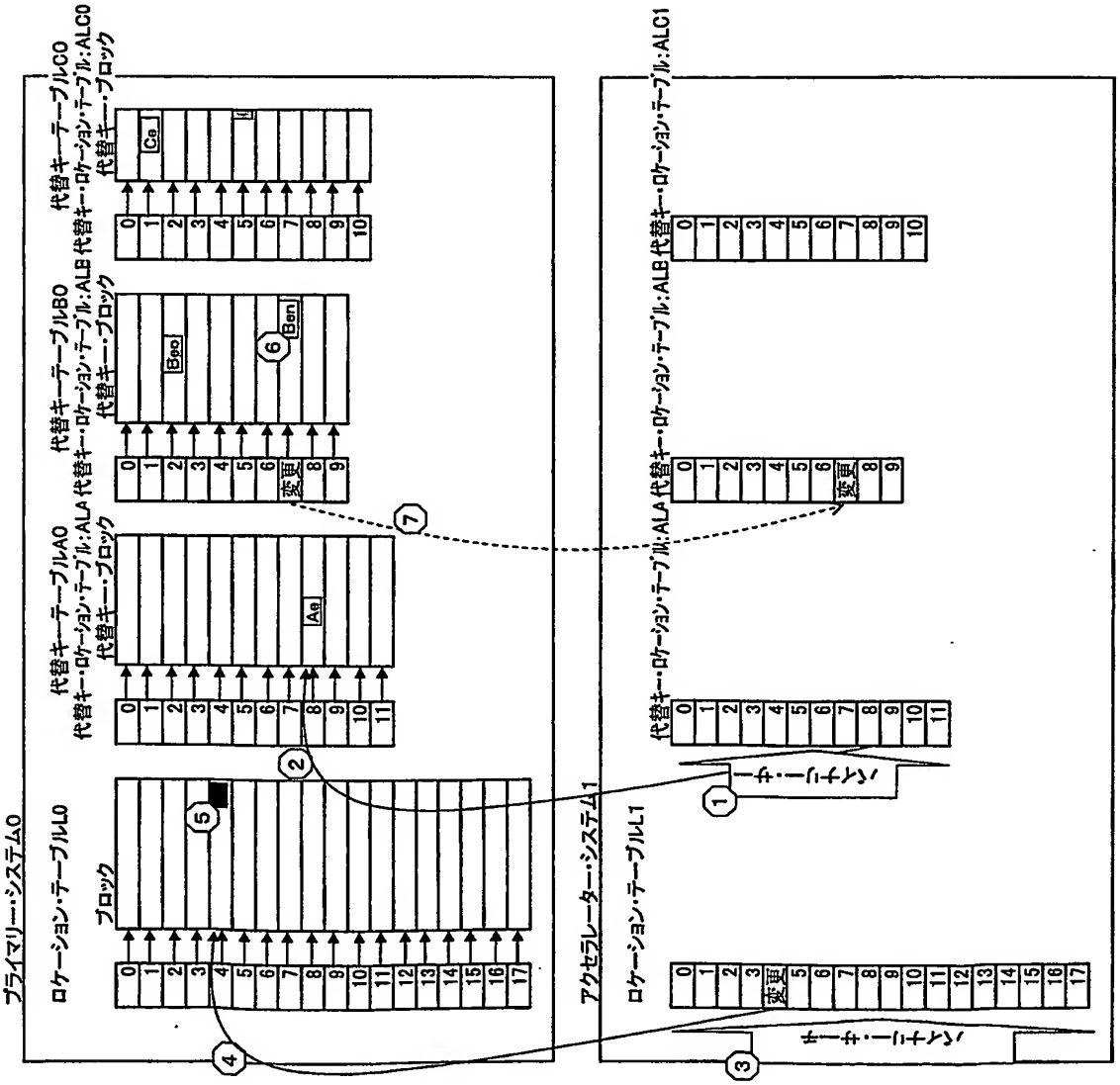

プライマリー・システム0
ロケーション・テーブルL0
ブロック
代替キー・ロケーション・テーブル:ALA
代替キーテーブルA0
代替キー・ブロック
代替キー・ロケーション・テーブル:ALB
代替キーテーブルB0
代替キー・ブロック
代替キー・ロケーション・テーブル:ALC
代替キーテーブルC0
代替キー・ブロック
変更
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
変更
インナリー・サーチ
インナリー・サーチ

図14

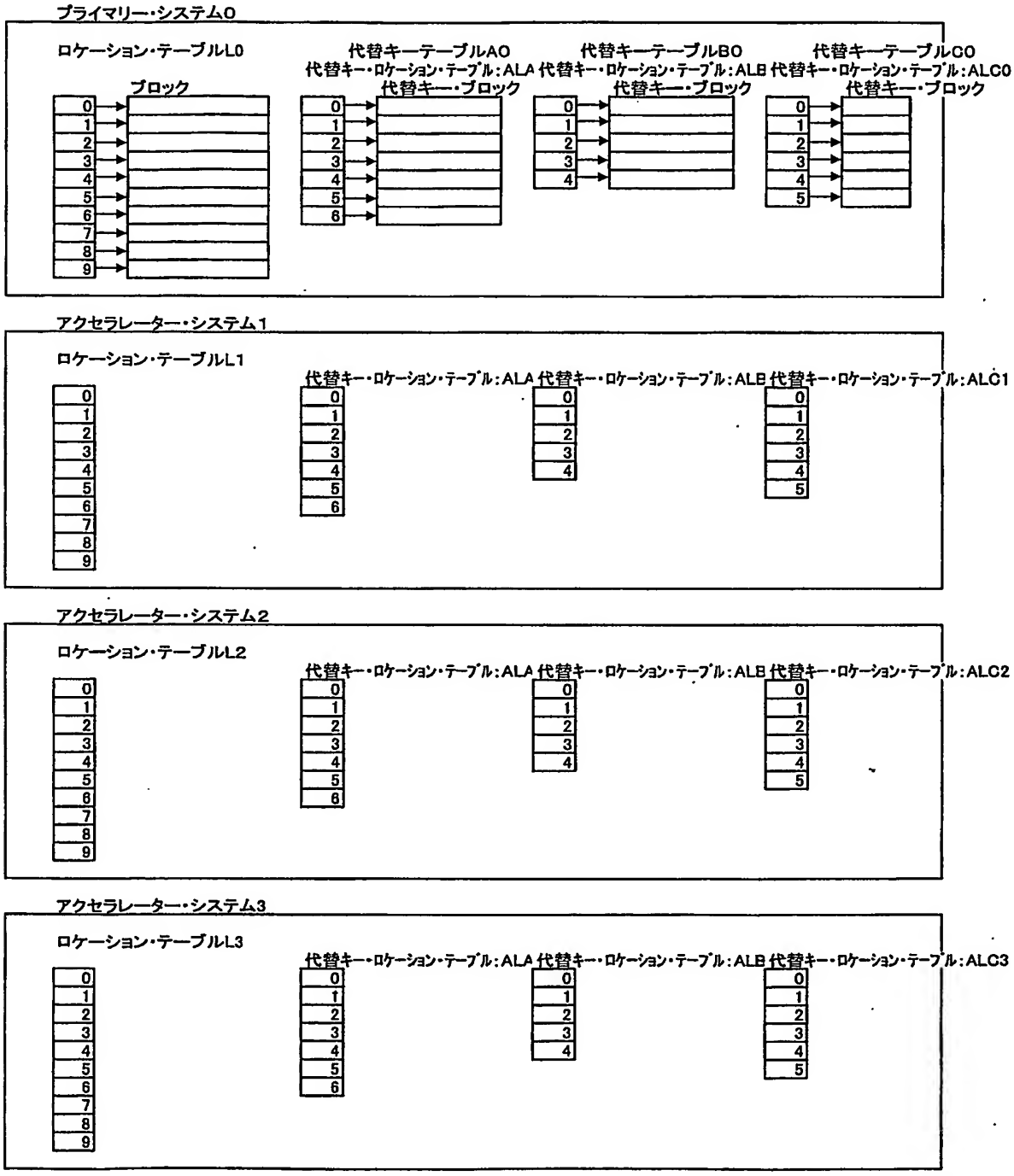

プライマリー・システム0
ロケーション・テーブルL0
ブロック
代替キーテーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キーテーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キーテーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
アクセラレーター・システム2
ロケーション・テーブルL2
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC2
アクセラレーター・システム3
ロケーション・テーブルL3
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC3

図15

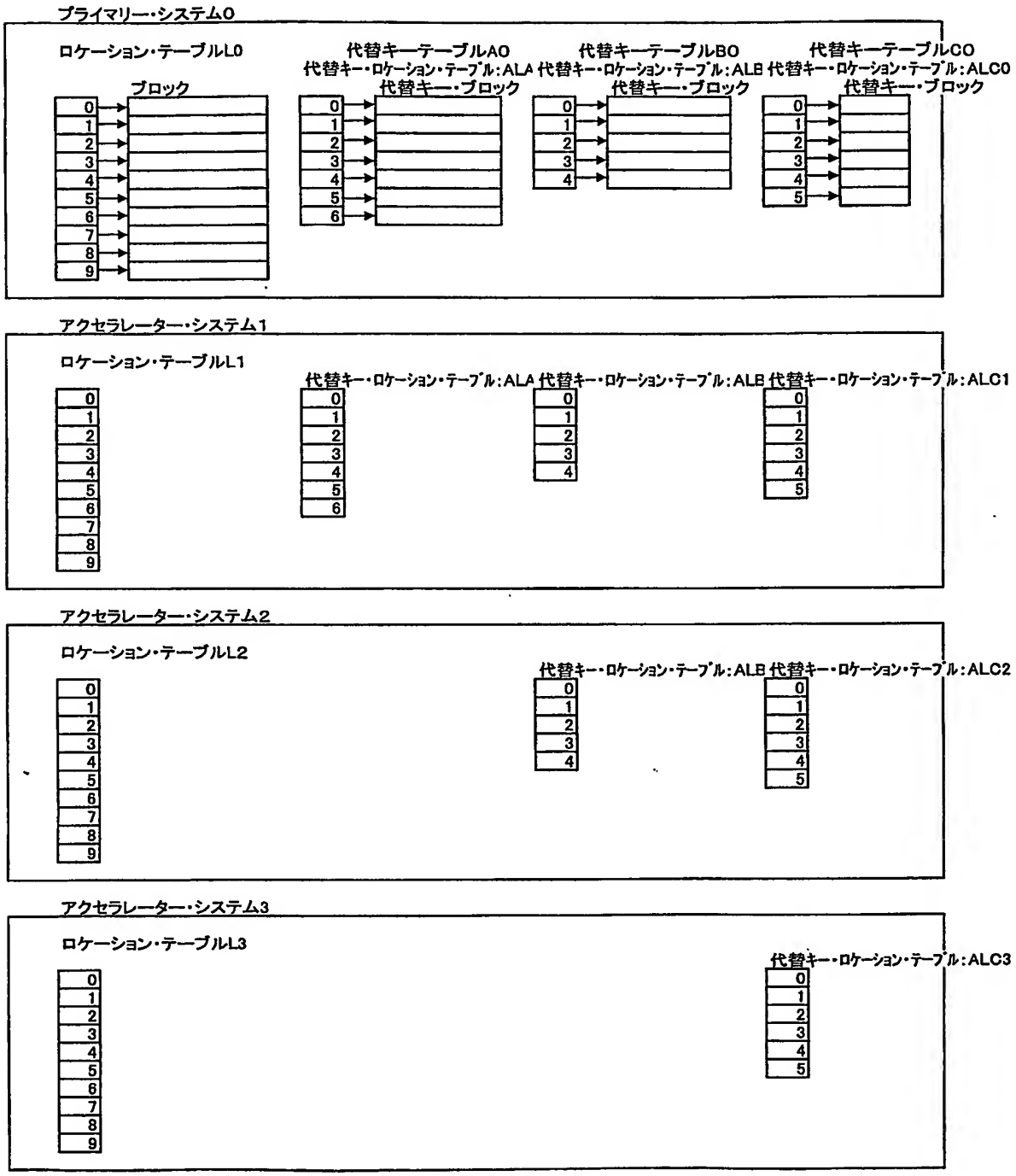

プライマリー・システム0
ロケーション・テーブルL0
ブロック
0
1
2
3
4
5
6
7
8
9
代替キーテーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キーテーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キーテーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
アクセラレーター・システム2
ロケーション・テーブルL2
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC2
アクセラレーター・システム3
ロケーション・テーブルL3
代替キー・ロケーション・テーブル:ALC3

図16

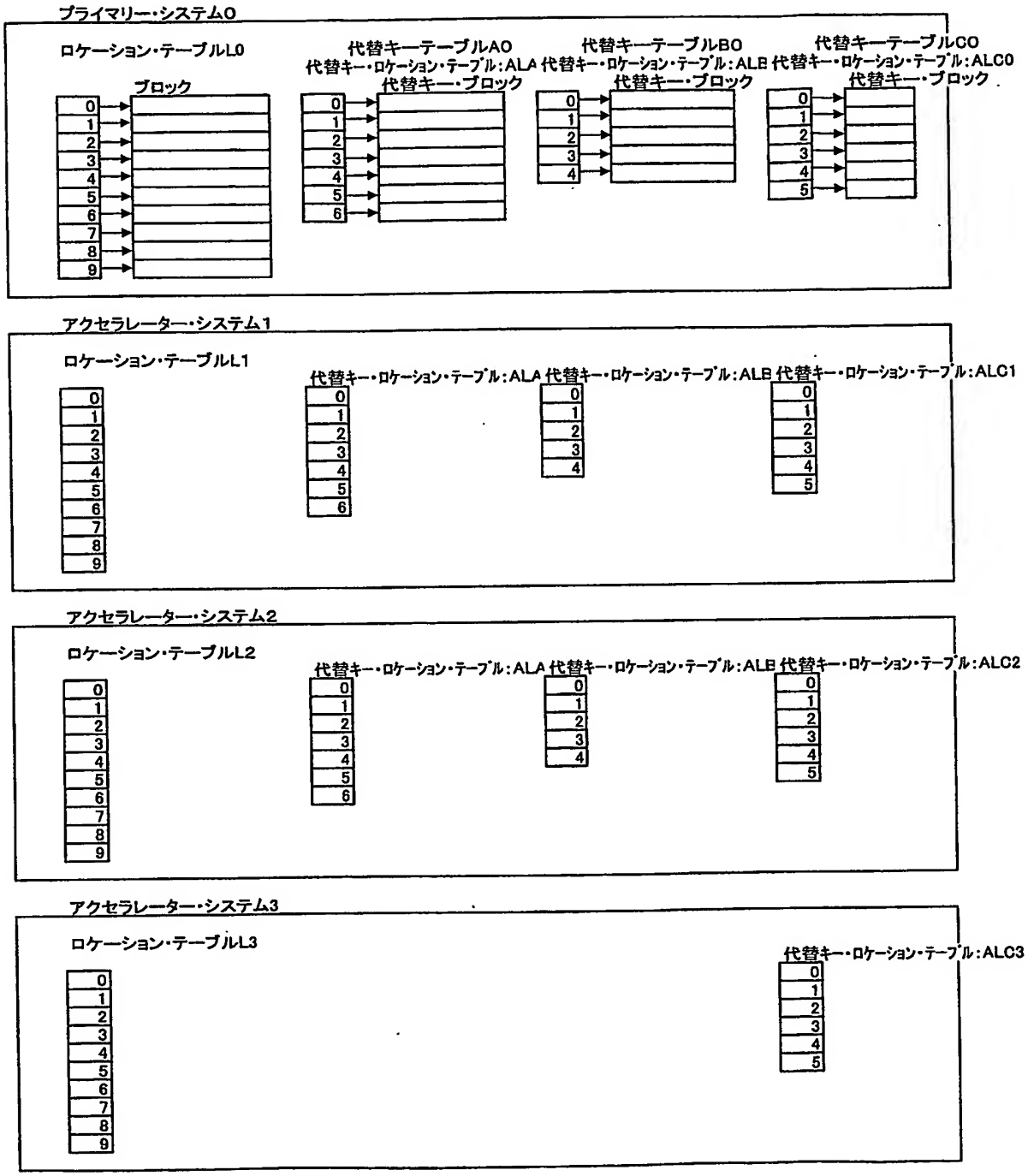
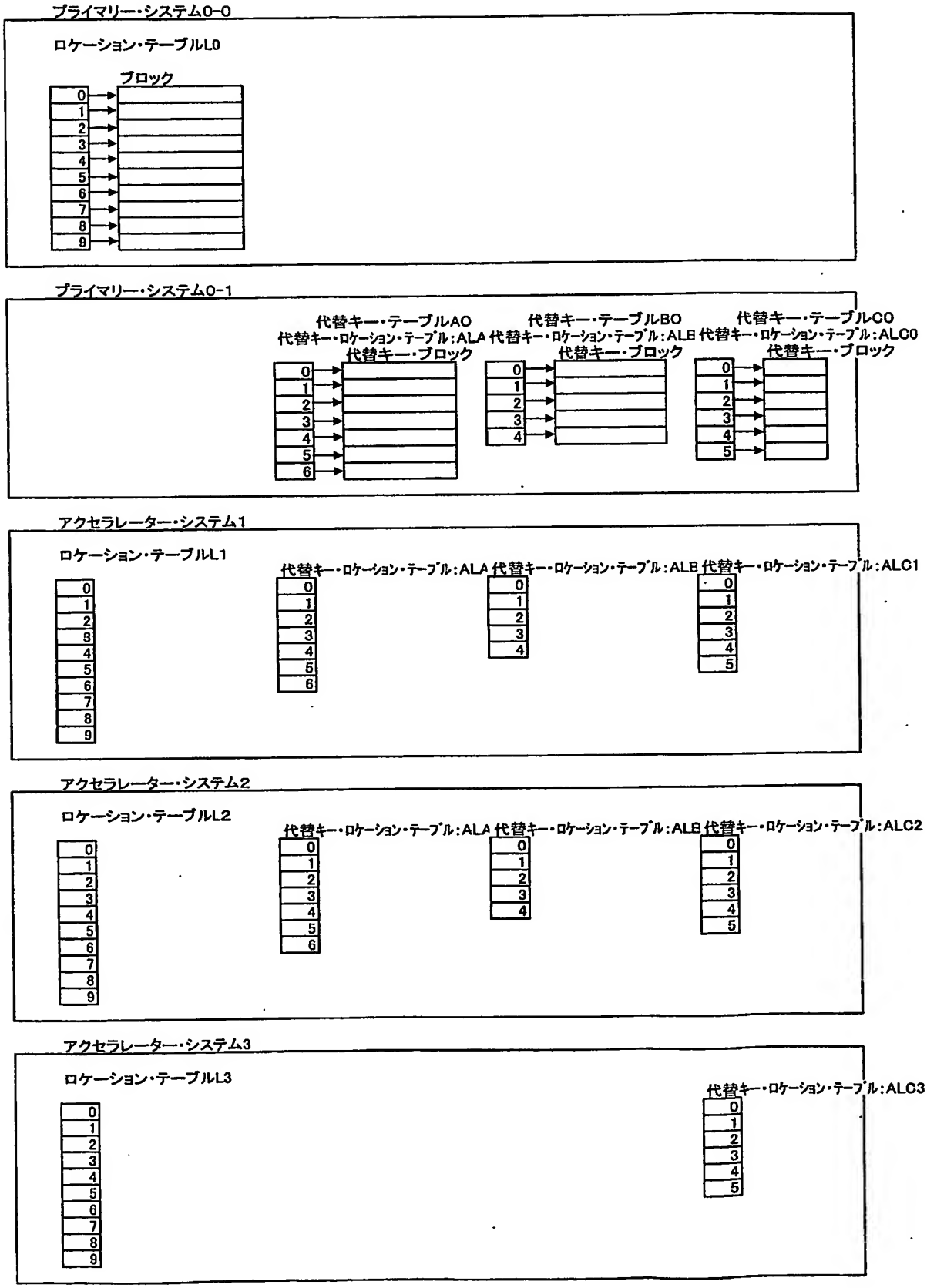

プライマリー・システム0
ロケーション・テーブルL0
ブロック
0
1
2
3
4
5
6
7
8
9
代替キーテーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キーテーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キーテーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
アクセラレーター・システム2
ロケーション・テーブルL2
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC2
アクセラレーター・システム3
ロケーション・テーブルL3
代替キー・ロケーション・テーブル:ALC3

図17


プライマリー・システム0-0
ロケーション・テーブルL0
ブロック
0
1
2
3
4
5
6
7
8
9
プライマリー・システム0-1
代替キー・テーブルA0
代替キー・ロケーション・テーブル:ALA
代替キー・ブロック
代替キー・テーブルB0
代替キー・ロケーション・テーブル:ALB
代替キー・ブロック
代替キー・テーブルC0
代替キー・ロケーション・テーブル:ALC0
代替キー・ブロック
アクセラレーター・システム1
ロケーション・テーブルL1
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC1
アクセラレーター・システム2
ロケーション・テーブルL2
代替キー・ロケーション・テーブル:ALA
代替キー・ロケーション・テーブル:ALB
代替キー・ロケーション・テーブル:ALC2
アクセラレーター・システム3
ロケーション・テーブルL3
代替キー・ロケーション・テーブル:ALC3

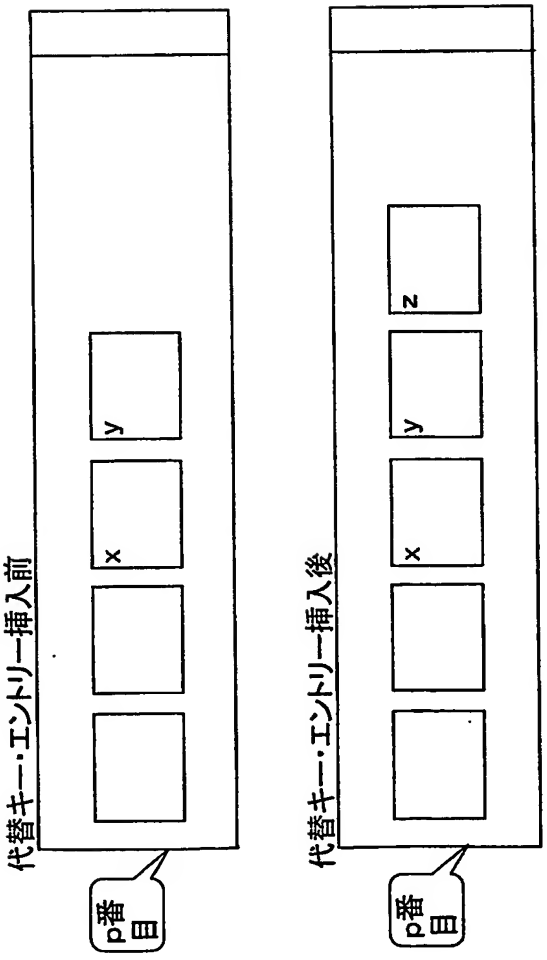
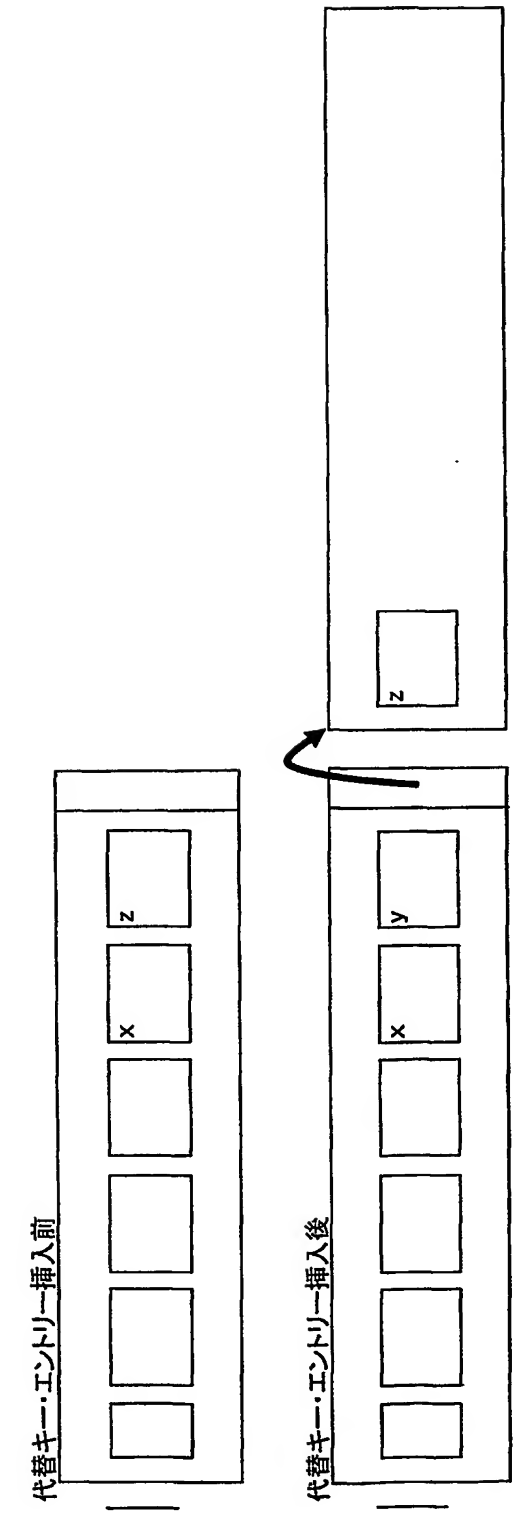


図18

図19



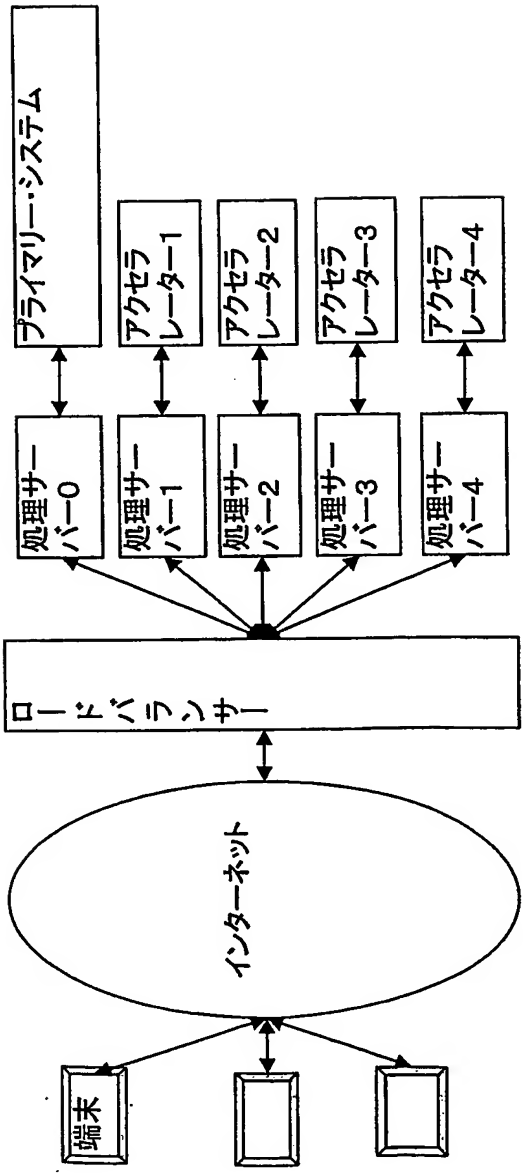

プライマリー・システム
アクセラレーター1
アクセラレーター2
アクセラレーター3
アクセラレーター4
処理サーバー0
処理サーバー1
処理サーバー2
処理サーバー3
処理サーバー4
ロードバランサー
インターネット
端末

図20

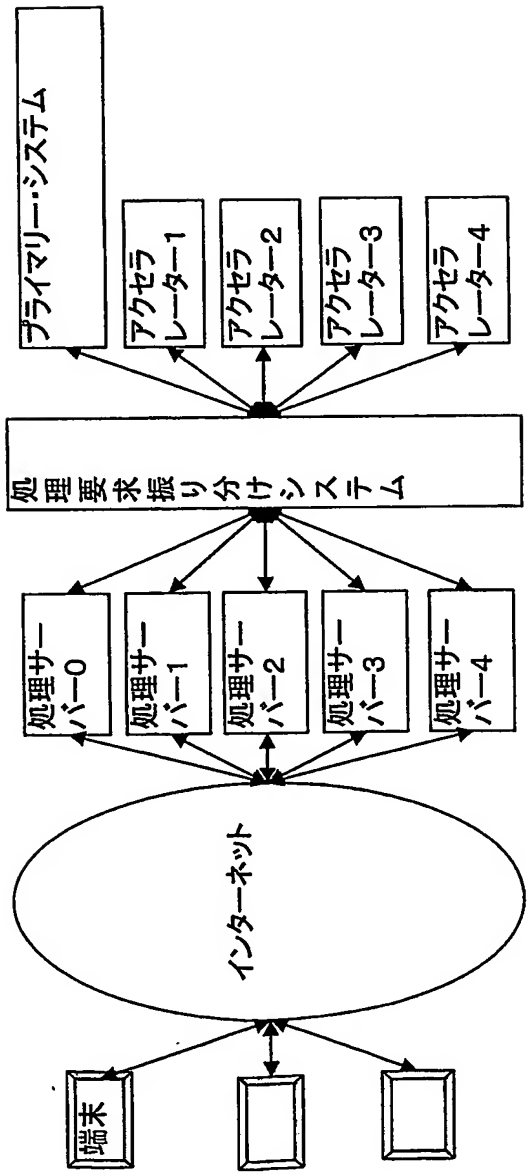
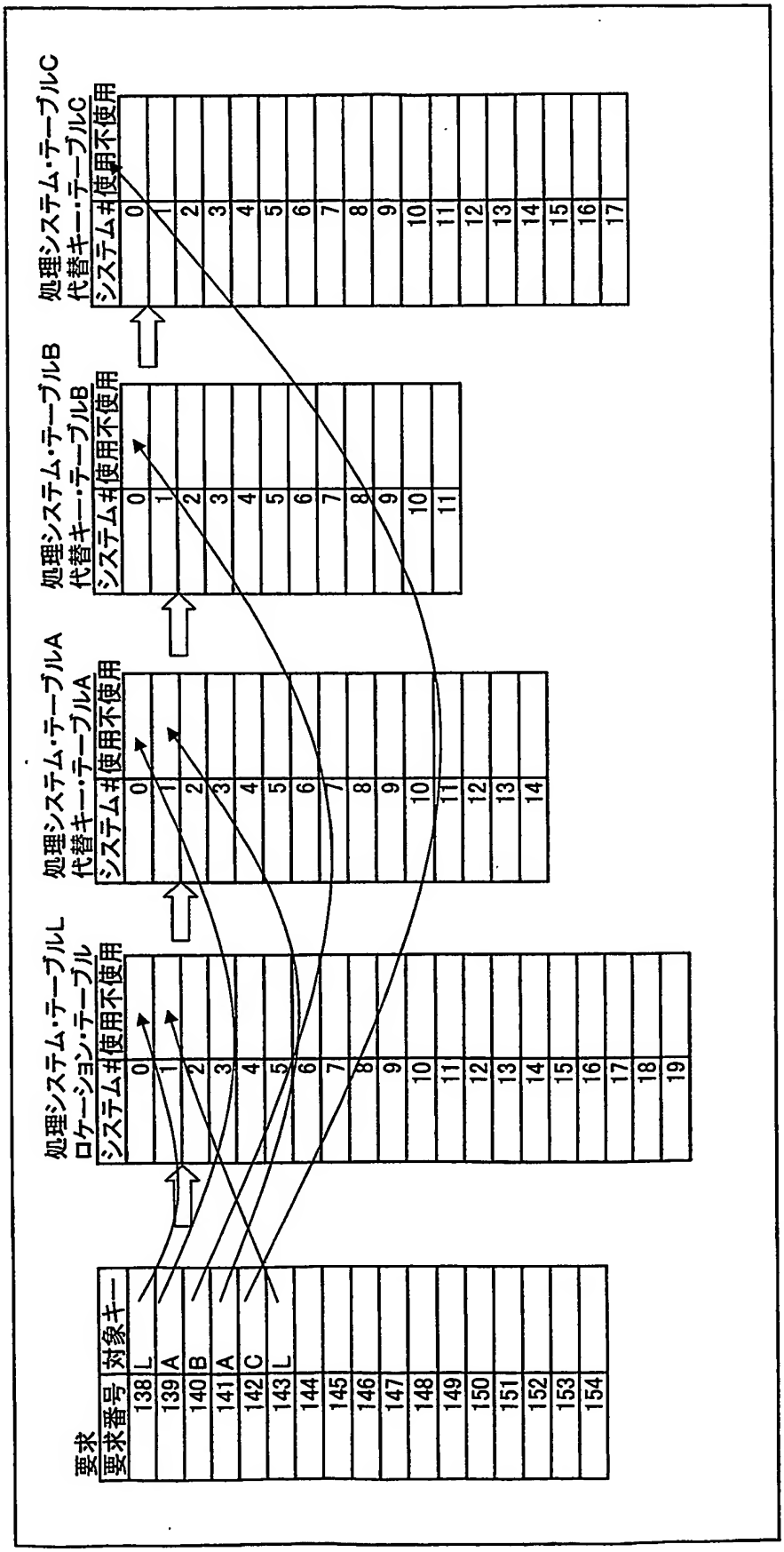

プライマリー・システム
アクセラレーター1
アクセラレーター2
アクセラレーター3
アクセラレーター4
処理要求振り分けシステム
処理サーバー0
処理サーバー1
処理サーバー2
処理サーバー3
処理サーバー4
インターネット
端末

図21

図22


要求
要求番号
対象キー
138 L
139 A
140 B
141 A
142 C
143 L
144
145
146
147
148
149
150
151
152
153
154
処理システム・テーブルL
ロケーション・テーブル
システム#
使用不使用
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
処理システム・テーブルA
代替キー・テーブルA
システム#
使用不使用
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
処理システム・テーブルB
代替キー・テーブルB
システム#
使用不使用
0 1 2 3 4 5 6 7 8 9 10 11
処理システム・テーブルC
代替キー・テーブルC
システム#
使用不使用
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

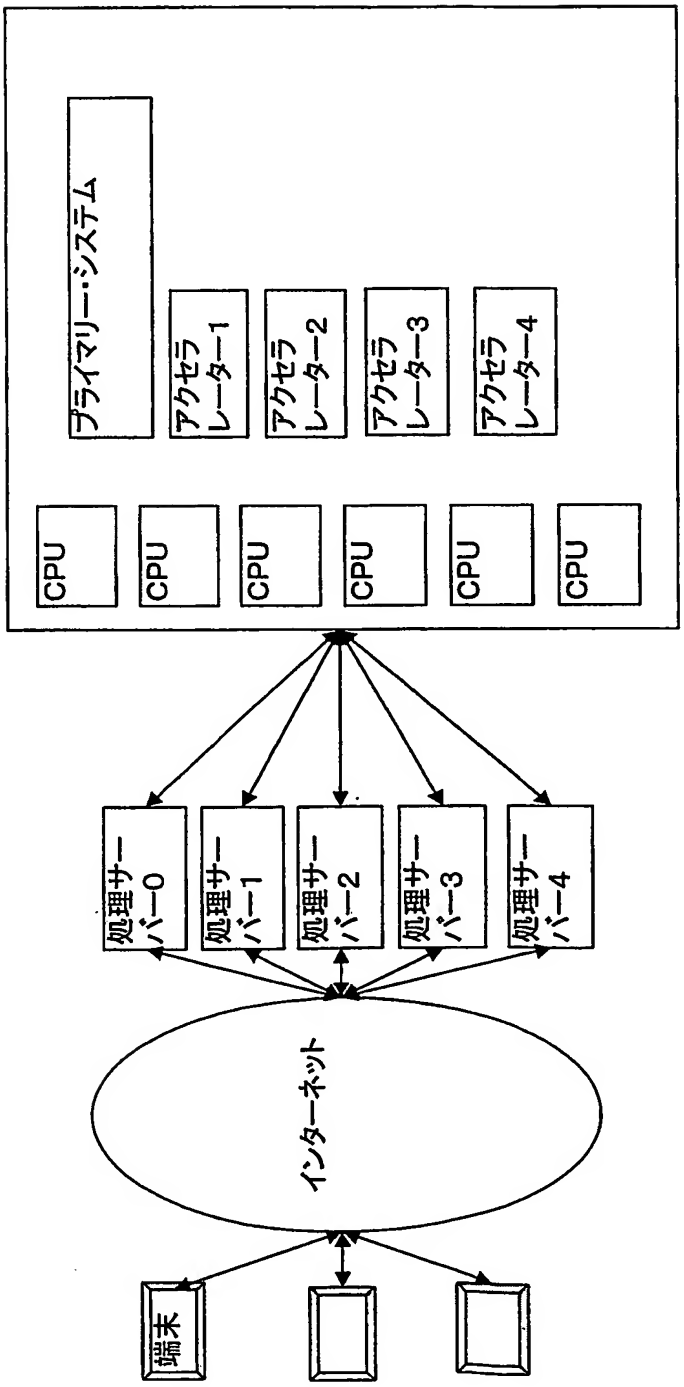

プライマリー・システム
アクセラレーター1
アクセラレーター2
アクセラレーター3
アクセラレーター4
CPU
CPU
CPU
CPU
CPU
CPU
処理サーバー0
処理サーバー1
処理サーバー2
処理サーバー3
処理サーバー4
インターネット
端末

図23

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/13443

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F12/00, G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F12/00, G06F17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2004
Kokai Jitsuyo Shinan Koho	1971-2004	Toroku Jitsuyo Shinan Koho	1994-2004

Electronic data base consulted during the inter national search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2000-181768 A (Annex Systems Inc.), 30 June, 2000 (30.06.00), Full text; Figs. 1 to 7 (Family: none)	1,4,6,8, 11-12
A	JP 60-103461 A (Fujitsu Ltd.), 07 June, 1985 (07.06.85), Full text; Figs. 1 to 2 (Family: none)	1,4,6,8, 11-12
A	JP 62-276626 A (Mitsubishi Electric Corp.), 01 December, 1987 (01.12.87), Full text; Figs. 1 to 3 (Family: none)	1,4,6,8, 11-12

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 29 January, 2004 (29.01.04)	Date of mailing of the international search report 10 February, 2004 (10.02.04)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/13443

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 5-334165 A (Hitachi, Ltd.), 17 December, 1993 (17.12.93), Full text; Figs. 1 to 17 & US 005515531 A	1-12

国際調査報告

国際出願番号　PCT／JP03／13443

A.　発明の属する分野の分類（国際特許分類（IPC））
Int. Cl⁷　G06F12／00, G06F17／30

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））
Int. Cl⁷　G06F12／00, G06F17／30

最小限資料以外の資料で調査を行った分野に含まれるもの
日本国実用新案公報　1922－1996年
日本国公開実用新案公報　1971－2004年
日本国実用新案登録公報　1996－2004年
日本国登録実用新案公報　1994－2004年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
A	JP 2000－181768 A（アネックスシステムズ株式会社）2000.06.30, 全文, 第1－7図（ファミリーなし）	1, 4, 6, 8, 11－12
A	JP 60－103461 A（富士通株式会社）1985.06.07, 全文, 第1－2図（ファミリーなし）	1, 4, 6, 8, 11－12
A	JP 62－276626 A（三菱電機株式会社）1987.12.01, 全文, 第1－3図（ファミリーなし）	1, 4, 6, 8, 11－12
A	JP 5－334165 A（株式会社日立製作所）1993.12.17, 全文, 第1－17図 & US 005515531 A	1－12

☐ C欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日　29. 01. 2004

国際調査報告の発送日　10. 2. 2004

国際調査機関の名称及びあて先
日本国特許庁（ISA／JP）
郵便番号100－8915
東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）　5N　9751
土田　行一

電話番号　03－3581－1101　内線　3545